NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

UNDERSEA NODE LOCALIZATION USING NODE-TO-NODE ACOUSTIC RANGES IN A DISTRIBUTED SEAWEB NETWORK

by

David C. Zinkhon

March 2009

Thesis Advisor: Joseph A. Rice

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.		
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** March 2009	**3. REPORT TYPE AND DATES COVERED** Master's Thesis
4. TITLE AND SUBTITLE Undersea Node Localization Using Node-to-Node Acoustic Ranges in Distributed Seaweb Network		**5. FUNDING NUMBERS**
6. AUTHOR(S) David C. Zinkhon		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.		
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Seaweb is a wide-area network interconnecting a set of distributed underwater nodes through the use of a DSP-based acoustic communications modem at each node and through-water digital acoustic links between neighboring nodes. As a by-product of Seaweb communications, the distances between neighboring nodes are obtained from the round-trip acoustic travel-time measurements. If the network is deployed in an ad hoc distribution, or if an established network is disturbed, the locations of the nodes are unknown to the operator. This thesis uses the node-to-node ranges, which have been compiled at the designated master node, as input to an algorithm for estimating the relative locations of all nodes. Synthetic network geometries serve to evaluate the algorithm with perfect ranges and with imperfect ranges and/or incomplete data. Seaweb networks deployed at sea are the final test of the algorithm.

14. SUBJECT TERMS underwater acoustics, Seaweb, localization, difference linearization			**15. NUMBER OF PAGES** 116
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

UNDERSEA NODE LOCALIZATION USING NODE-TO-NODE ACOUSTIC RANGES IN A DISTRIBUTED SEAWEB NETWORK

David C. Zinkhon
Lieutenant, United States Navy
B.S., University of New Mexico, 2002

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING ACOUSTICS

from the

NAVAL POSTGRADUATE SCHOOL
March 2009

Author:	David C. Zinkhon
Approved by:	Joseph A. Rice Thesis Advisor
	Daphne Kapolka Chair, Engineering Acoustics Academic Committee

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Seaweb is a wide-area network interconnecting a set of distributed underwater nodes through the use of a DSP-based acoustic communications modem at each node and through-water digital acoustic links between neighboring nodes. As a by-product of Seaweb communications, the distances between neighboring nodes are obtained from the round-trip acoustic travel-time measurements. If the network is deployed in an ad hoc distribution, or if an established network is disturbed, the locations of the nodes are unknown to the operator. This thesis uses the node-to-node ranges, which have been compiled at the designated master node, as input to an algorithm for estimating the relative locations of all nodes. Synthetic network geometries serve to evaluate the algorithm with perfect ranges and with imperfect ranges and/or incomplete data. Seaweb networks deployed at sea are the final test of the algorithm.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First I would like to thank my wife, Amy, for the past eleven years of continued love, support, and inspiration, as well as my two beautiful daughters, Grace and Elizabeth.

To my thesis advisor, Professor Joe Rice, for mentoring through the thesis process and allowing me to work with the technologies and opportunities Seaweb presents.

To Chris Fletcher of SPAWAR Systems Center Pacific, for his expeditious retrieval of vital data.

To the numerous professors and staff that made my time at the Naval Postgraduate School the enriching experience it was.

And finally to my fellow students, Chris, Paul, Nevin, Greg, Dave, Chee Wei, Bill, and Jason that helped me through the challenging classes and listened to my ideas.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PROBLEM STATEMENT

Seaweb is an acoustic communications technology enabling underwater sensor networks. Data packets usually flow from a sensor node to the master node, with command and control packets flowing in the reciprocal path. More generally, a network packet begins at the source node, travels through repeater nodes, and ends at the destination node. The network routes are dictated by routing tables stored in the acoustic modems. Prior to Ong's development of the Seaweb ad hoc discovery process [1], the routing tables were initialized and maintained by the operator. By using the algorithms developed in [1], Seaweb networks are now capable of self-determining the routing tables, thereby allowing the network to be deployed with an ad hoc distribution. However, with ad hoc deployments, the operator may not know the locations of the nodes within the network. When the routing algorithm from [1] is implemented, the node-to-node ranges are measured through the use of acoustic propagation time delays and assumptions about the speed of sound in water.

Previous thesis work by Hahn [2] and Ouimet [3] used acoustic communication ranging in the Seaweb network for localizing unmanned underwater vehicles (UUVs). Respectively, their methods involved weighted solutions and center of mass approaches. Later, Reed [4] used the method of difference linearization, based on the principles used in the Global Positioning System (GPS), to produce superior results for tracking UUVs.

This thesis uses the formulas developed in [4] to estimate the relative positions of the nodes using planar trigonometry.

B. SCOPE

1. Objective of Algorithm

The algorithm produced in this thesis uses the range data measured during the Seaweb discovery process to estimate the horizontal positions of the discovered nodes.

This is accomplished by first setting up a horizontal coordinate system that can be used as a frame of reference. The origin is fixed at the location of the master node because it is usually able to self locate by means of GPS. The positive *x*-axis is oriented to intersect with a neighbor node of the master. This node is referred to as the 'on_X' node and is preferably located along a known line of bearing from the 'master' node to allow the end user to translate the *x*-*y* coordinate system to a traditional latitude-longitude grid.

Next, range data gathered each time a node conducts a broadcast ping are organized into a time-stamped $N \times N$ matrix. These matrices are compiled into a stack that is sequenced by the time-stamp. The algorithm then looks down each column of the stack and statistically analyzes all the ranges available between a particular pair of nodes, eliminating any ranges that fall outside a ten percent confidence interval from the mean. Next the algorithm uses the filtered mean ranges and localizes the node positions using the formulas outlined in [4]. Due to ambiguous solutions when insufficient ranges are known for a particular node, several solution sets may be possible. The output of the algorithm is the *x*-*y* position of each node and a visual representation of each ambiguous solution.

2. Limitations Not Addressed

While the algorithm estimates the relative horizontal locations of the network nodes, it does not attempt to then rectify these locations to the traditional latitude-longitude coordinate system. The algorithm also does not attempt to correct for range errors inherent to the method used to derive them; it simply eliminates statistical outliers from the range data in an attempt to find the best solution given all the data.

C. OUTLINE OF THESIS

Chapter II is an overview of Seaweb equipment and operations, including the acoustical method used to measure inter-node ranges. Chapter III then addresses some of the errors inherent with this ranging method. Chapter IV discusses the localization methods used by the algorithm, and Chapter V details their implementation. Chapter VI describes results obtained with various synthetic data, including error-free as well as incomplete and/or inaccurate data. Chapter VII analyzes the range data from Seaweb sea trial testing. Chapter VIII summarizes the conclusions reached in the thesis, and Chapter IX discusses recommendations and possible further research.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

A. SEAWEB OPERATIONS

Seaweb is a networking technology supporting underwater communication between any number of fixed sensor nodes, repeater nodes, and gateway nodes. It is also configurable to communicate with mobile nodes including submarines and UUVs [5, 6]. Communications through the water with only a single gateway node at the sea surface improves both the stealth and survivability of the network. A representative deployment of Seaweb is shown in Figure 1, where the network includes a submerged UUV, a radio-acoustic communication (racom) buoy acting as the gateway node, and repeater nodes. The UUV uses the network infrastructure while submerged, and the UUV itself can act as a gateway node while on the surface.

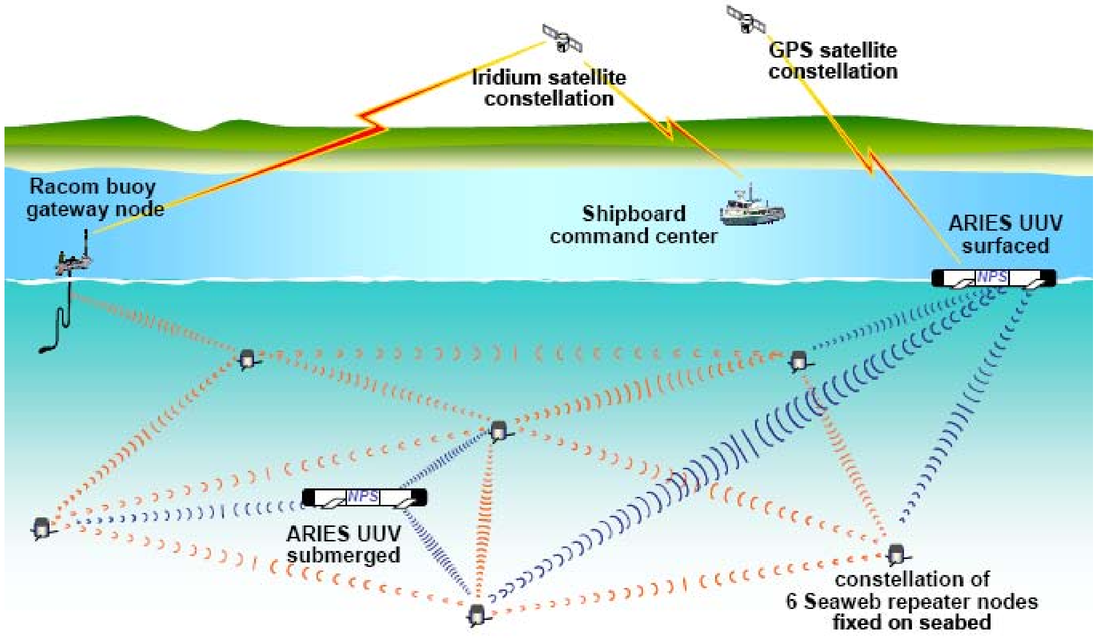

Iridium satellite constellation
GPS satellite constellation
Racom buoy gateway node
Shipboard command center
ARIES UUV surfaced
NPS
NPS
ARIES UUV submerged
constellation of 6 Seaweb repeater nodes fixed on seabed

Figure 1. Typical Seaweb configuration consisting of both fixed nodes mounted on the seabed along with UUVs acting as mobile nodes (from [2]).

1. Equipment

The physical composition of Seaweb consists of three components: the Seaweb server, underwater sensor/repeater nodes, and a gateway buoy that allows interactions between the server and underwater nodes. The Seaweb server is physically located either onboard a control ship or ashore, the only requirement being that radio communications with the gateway buoy are available. If direct radio communications are not possible with the gateway buoy, demonstrated alternatives are an Iridium satellite link or cellular telephone modem. The Seaweb server is used by the operator to command, control and monitor the deployed network [7].

The racom gateway buoy is moored at a convenient location within the Seaweb domain, either near the center to maximize the number of underwater nodes that it is in direct contact with, or at the edge to ensure a line-of-sight radio link to the Seaweb server. The buoy is maintained in place by an anchor and swivel system moored on the ocean floor. The surfaced portion of the buoy consists of solar panels, radio communication equipment, satellite communication equipment, and a GPS receiver. Seaweb acoustic communications occur in the 9-14 kHz frequency band through a submerged acoustic transducer married to the mooring line. Figure 2 depicts the racom gateway buoy configuration used at the St. Margaret’s Bay sea trials.

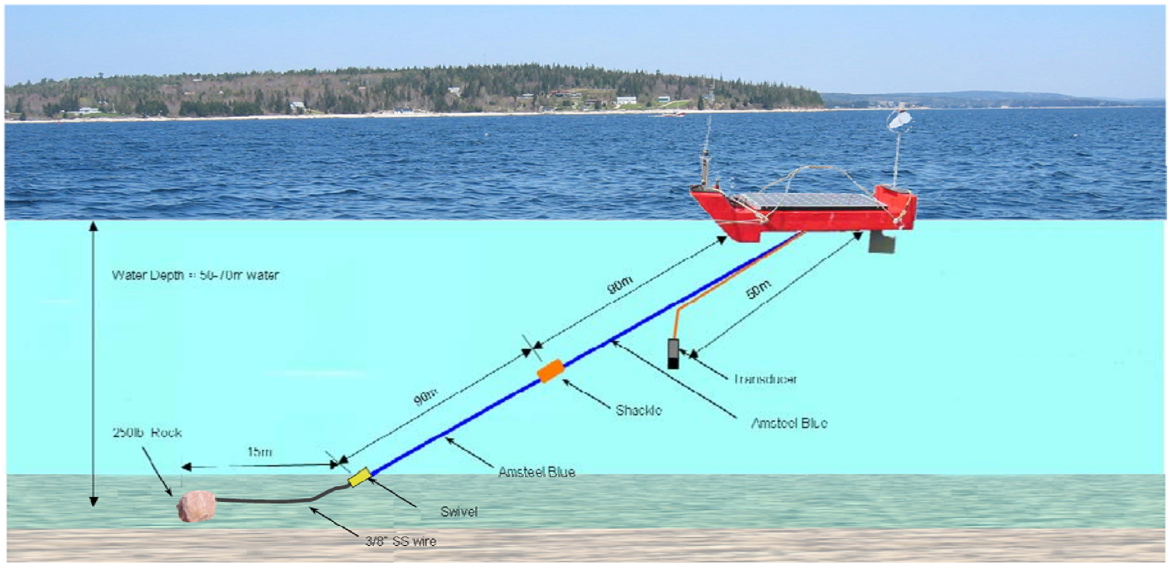


Figure 2. Racom gateway buoy (from [1]).

Underwater fixed nodes are generally sensor nodes or repeater nodes. The repeater nodes are moored to the seabed and usually consist of anchor, acoustic release module, telesonar modem, and subsurface float. The telesonar modem is commercially available from Teledyne Benthos, Inc. and is programmed with specialized Seaweb networking software and firmware [8]. The acoustic release serves for recovery of the repeater node. Figure 3 shows a typical deployed configuration of a repeater node.

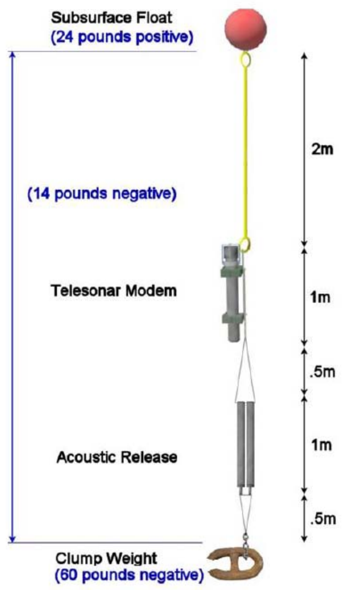


Figure 3. Seaweb repeater node using a telesonar modem (from [5]).

2. Network Layout

Seaweb networks can consist of any number of underwater nodes and gateway buoys, and the network routes are reconfigurable if a node failure occurs. Most Seaweb layouts consist of a single gateway node that is positioned in locations to maximize its survivability, i.e., out of ship traffic lanes and away from other near shore hazards. Often the gateway buoy is identified as the ‘master’ node, since it is in direct radio contact with the server. The term ‘master’ node is also used in the localization process, and since the gateway buoy position at the sea surface is known from GPS, it provides a geographic reference point. Care must be taken when deploying the first underwater node since this is the second reference point used by the localization algorithm, known as the ‘on_X’ node in the localization algorithm. The 'on_X' node should be deployed along a known line of bearing from the 'master' node thereby allowing the operator to fix the location of these two nodes on a nautical chart.

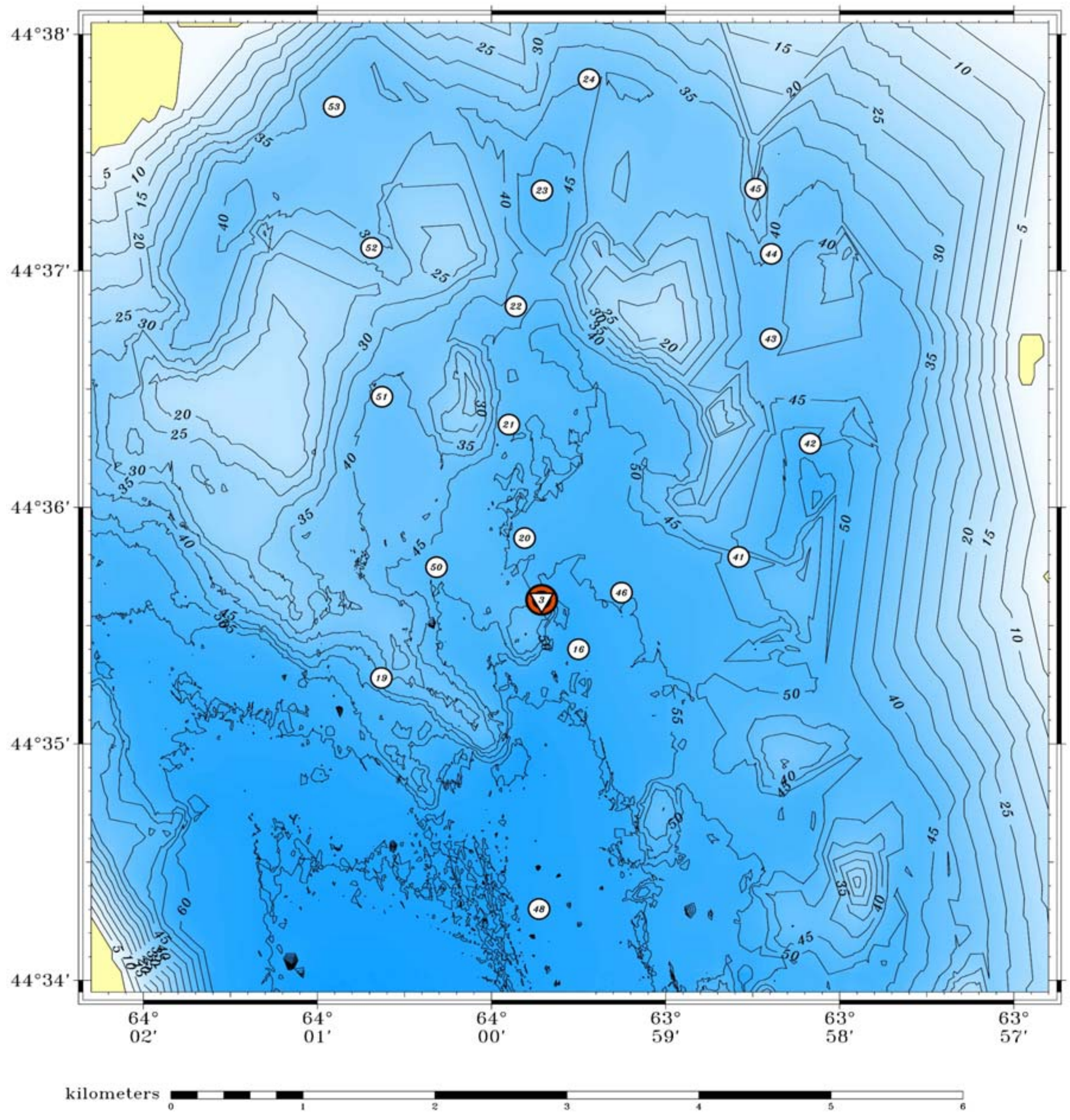


Figure 4. Location of Seaweb nodes in June 2008 at St. Margaret's Bay sea trial. Node 3 (in red) was the gateway buoy and master node (from [1]).

The rest of the network nodes are positioned in the area of interest. Typically, the layout is designed such that multiple communication routes are available for data transfer to each node. This increases the overall reliability of the network by producing redundant routes. The distance between the nodes can be expected to be one to five kilometers depending on the maximum acoustic communications range for the deployment environment. Significant environmental factors include the sound-speed profile, the background noise level, and transmission loss [9]. Figure 4 shows the network layout used in the St. Margaret's Bay sea trial in June 2008.

3. Link-Layer Protocols

Seaweb node-to-node communications use a link-layer protocol that allows for addressing, power control, adaptive modulation, and ranging [10]. The protocol uses a request to send (RTS) and clear to send (CTS) to establish first contact between two nodes prior to transferring data between the nodes. Any corrupted data are requested to be resent by the receiving node using a selective automatic repeat request (SRQ). This concept is diagrammed in Figure 5. Following the same handshake strategy, Seaweb also supports a PING/ECHO protocol useful during the ad hoc discovery process.

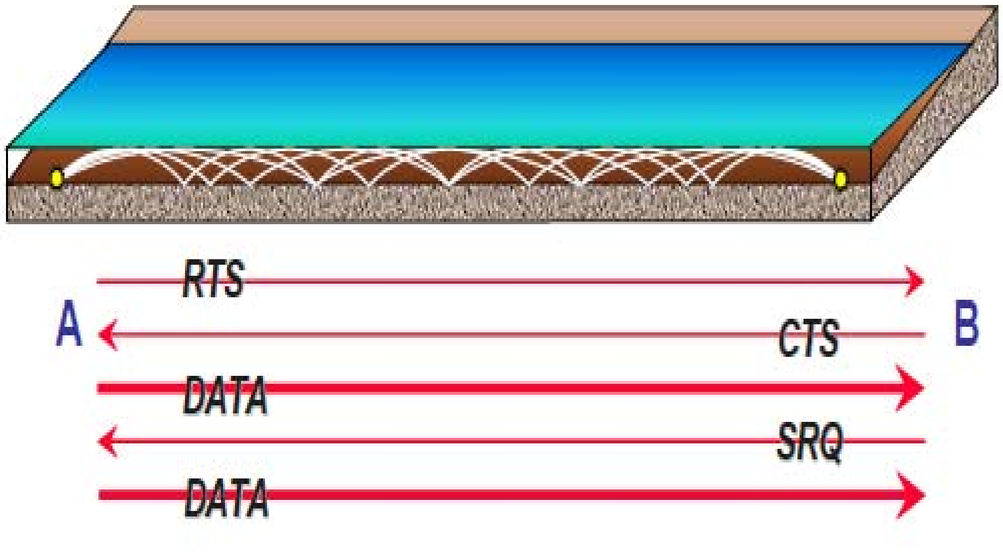


Figure 5. Seaweb node-to-node communication scheme (from [10]).

To determine the range between neighboring nodes, a hyperbolic frequency-modulated (HFM) chirp precedes the PING packet sent by the first node (node *i* in Figure 6, below). Node *j* receives the ping and determines the time of arrival by picking the peak of a HFM matched filter. Node *j* then waits a specified dwell time, τ_j, prior to issuing a return echo (with HFM) that follows the same path as the initial ping based on reciprocity. Therefore, the reciprocal sound propagation delays between nodes *i* and *j* are equal,

$$d_{ij} = d_{ji} \,. \tag{2.1}$$

Once the echo is received at node *i*, the Seaweb modem measures the range between the two nodes based on the total delay time,

$$t_j - t_o = d_{ij} + d_{ji} + \tau_j \,, \tag{2.2}$$

and substituting Equation (2.1) and solving for d_{ij} gives

$$d_{ij} = \frac{t_j - t_o - \tau_j}{2} \,. \tag{2.3}$$

Seaweb uses the assumption that the speed of sound in the ocean is c_o = 1500 m/s for determining the node-to-node range,

$$r_{ij} = c_o \times d_{ij} \,. \tag{2.4}$$

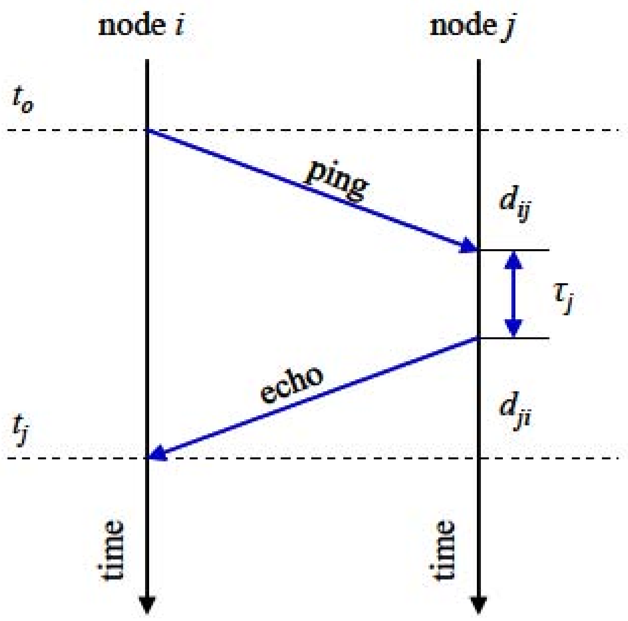


Figure 6. Seaweb ranging process (from [2]).

B. AD HOC DISCOVERY PROCESS

1. Broadcast Ping

Prior to the introduction of the ad hoc discovery process developed in [1], the routing tables were manually determined and maintained by the operator located at the Seaweb server. After a Seaweb network was deployed in the water, the first operation performed was node-to-node ranging to verify that the pre-programmed routing tables were functional. The Seaweb server initiated this process by directing the 'master' node to transmit a broadcast ping. When the various repeater nodes detect the broadcast ping, they reply with an echo that confirms that they are within acoustical communications range, the ranges are calculated, and then communicated back to the server. This process is depicted in Figure 7. The operator then sequentially directed each node to repeat the broadcast ping process until all nodes were discovered and all ranges reported to the server. All of this served only to confirm the validity of pre-programmed routes.

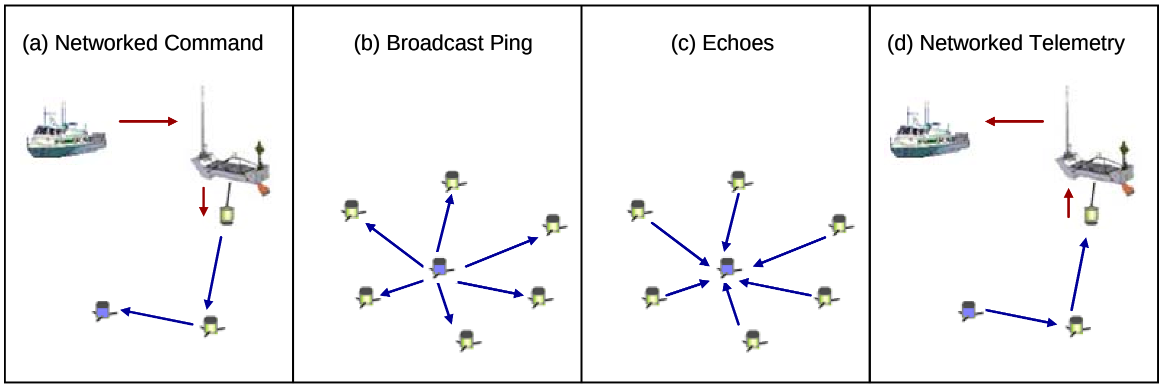


Figure 7. The broadcast ping process (after [11]).

2. Discovery

The discovery process developed in [1] automates the initialization of network routes. A summary of this process is shown in Figures 8, 9, and 10. First the master node conducts a broadcast ping, and then the master node systematically commands each discovered node to conduct its own broadcast ping to potentially discover more nodes. The master node continues to direct all new nodes to conduct their own broadcast ping.

Upon completion, the master node algorithm determines the optimum route to each node according to a cost function that considers the number of nodes that are in the path and the ranges between each node. A by-product of this process are data tables containing the ranges that each broadcast ping measured. The algorithm developed in this thesis uses these range data tables to locate the network nodes.

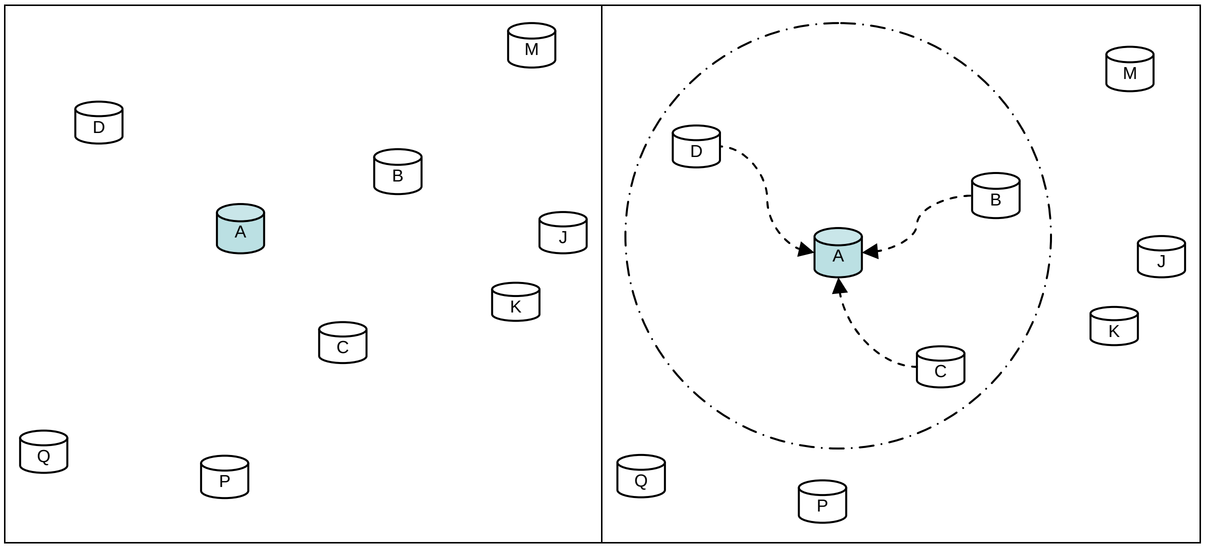


Figure 8. Left: Ad hoc locations of nodes. Right: Initial broadcast ping by master node, A, which discovers nodes B, C, and D (after [1]).

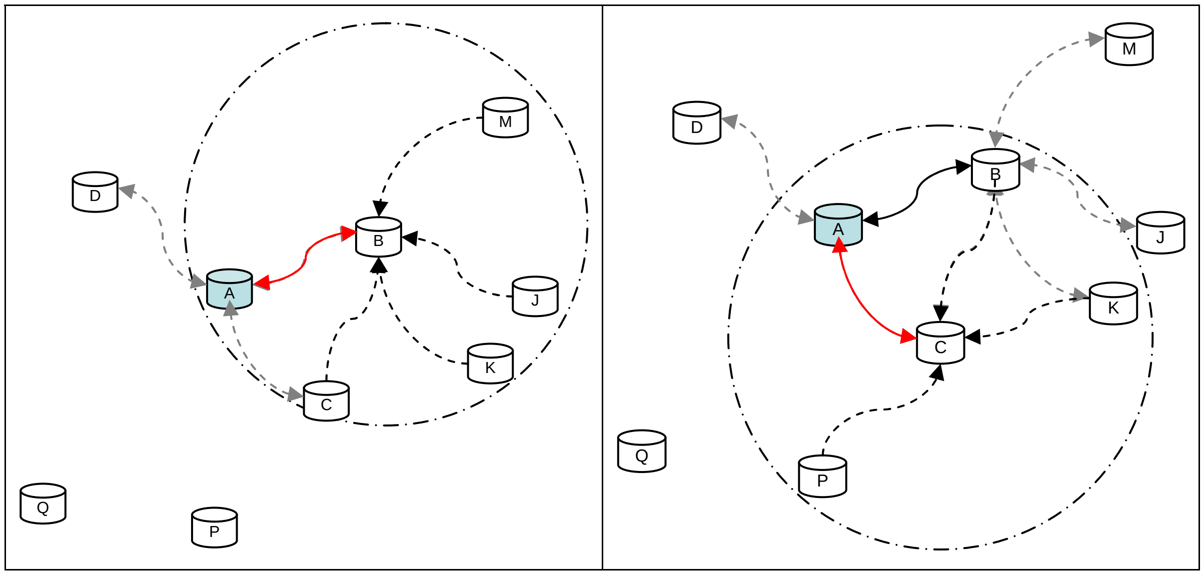


Figure 9. Left: Secondary broadcast pings by node B that discovers M, J, K, and determines range from B to C. Right: Secondary broadcast ping by C that discovers node P and determines ranges from C to B and C to K (after [1]).

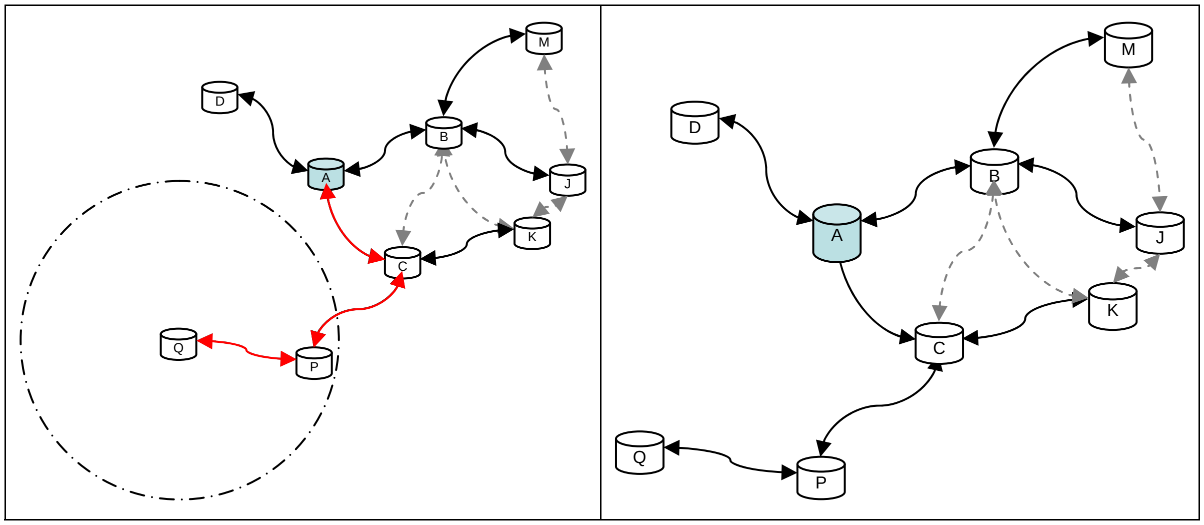


Figure 10. Left: tertiary ping by P that discovers node Q. Right: Depiction of routes to all nodes (after [1]).

THIS PAGE INTENTIONALLY LEFT BLANK

III. SOURCES OF ERROR

A. ACOUSTIC VARIABILITY

1. Sound Speed Profile

Two significant assumptions are made in the Seaweb estimation of node-to-node ranges, the first of which is that the sound speed, c_o, used in Equation (2.4) is 1500 meters per second. Sound speed in the ocean varies from 1480 m/s to 1520 m/s [12], which represents a $\pm 1.3\%$ deviation from the value assumed by Seaweb. For example, if the true sound speed is 1480 m/s and the range between two nodes is 3000 meters, the reported range would be 3040 meters. Assuming that the true sound speed is constant over the entire area of the network, this results in range solutions which are typically larger than the true ranges between the nodes.

The second assumption made by the Seaweb range estimation is that sound travels in straight lines and hence is linearly related to range. However, the variations in temperature, pressure, and salinity in the water column cause significant variability in the speed of sound. This results in refraction as sound passes through the various layers of the water column. To illustrate this, a representative sound speed profile is produced using Munk's canonical profile [13]

$$c(z) = c_0\left[1+\varepsilon\left(e^{-\eta}+\eta-1\right)\right] \tag{3.1}$$

where

$$\eta = \frac{2\left(z - z_{axis}\right)}{B} \tag{3.2}$$

and c_o is the sound speed at the sound channel axis, ε is the perturbation coefficient, z_{axis} is the depth of the sound channel axis, B is the scale depth, and η is the dimensionless distance beneath the sound channel axis. For this example the values of the constants are chosen as $z_{axis} = 1000\text{ m}$, $B = 1000\text{ m}$, $\varepsilon = 0.0057$, $c_o = 1500\text{ m/s}$, and the maximum depth is 4500 m. The resultant sound speed profile is depicted in Figure 11.

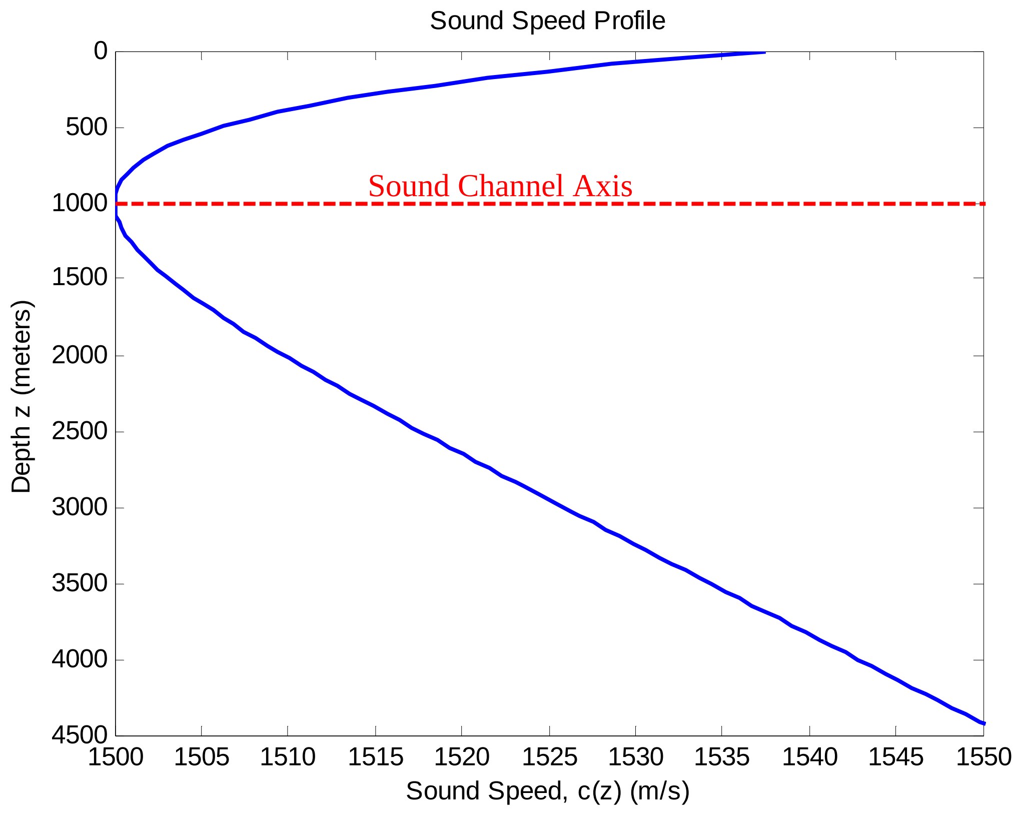
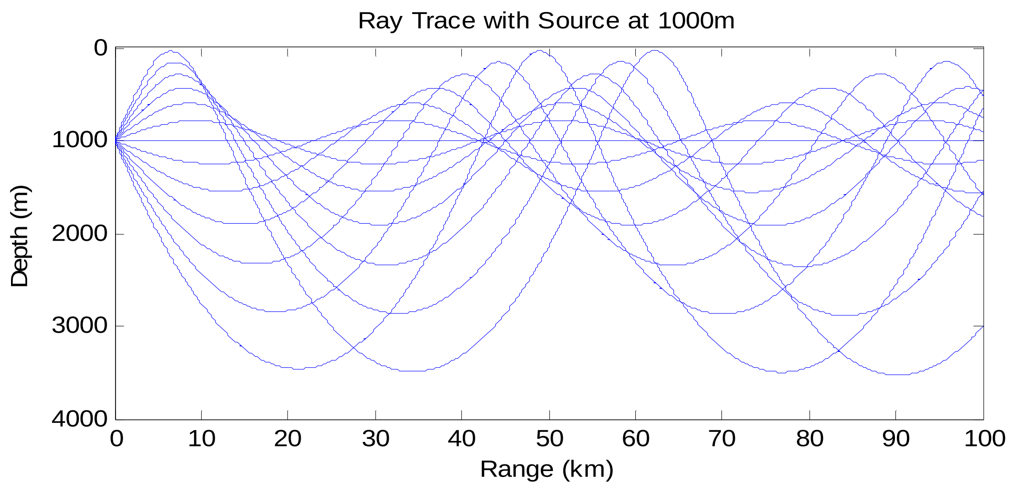


Figure 11. Representative sound speed profile based on Munk’s canonical profile.

If the source and receiver nodes are located at the axis of the sound speed profile, a straight non-refracted path between them exists. When either source or receiver is not located at the sound channel axis, there is no straight line path between them. This is graphically illustrated in Figure 12 using fifteen rays (note: if a ray strikes either the surface or the bottom, it is removed from the analysis). Since the actual ray paths can be longer than the straight line path assumed by the Seaweb ranging estimation, the measured ranges tend to be overestimates of the actual ranges. Using the constraints of this example, for a 5000 meter distance between nodes, the travel time difference could be as great as 0.1 second which equates to a range overestimate of 150 meters, which is an error of 3%.



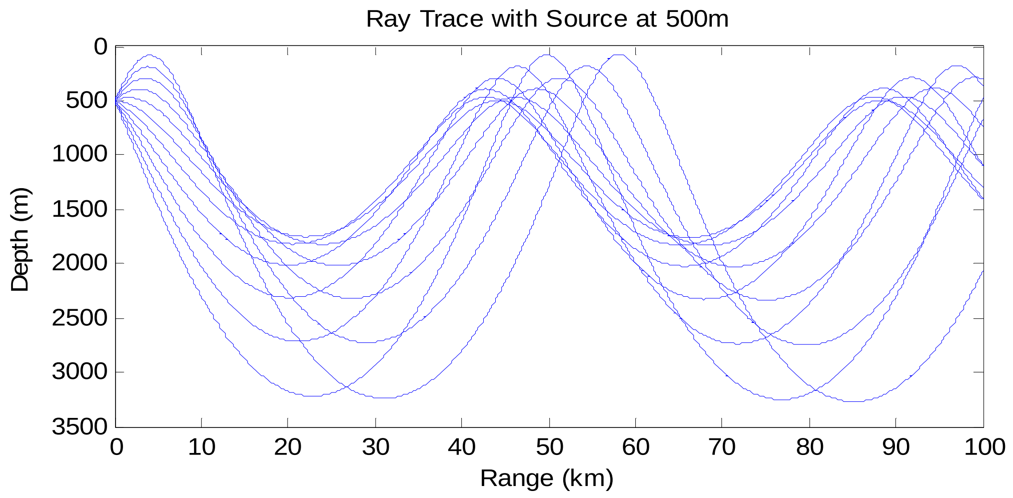


Figure 12. Ray traces for source located at the sound channel axis (top) and above the sound channel axis (bottom).

2. Reflected Paths

The previous discussion does not account for rays that interact with either the surface or the bottom, both of which occur in the real world. Interactions with a boundary results in absorption and scattering that reduce the intensity of the sound energy. For the surface, the amount of reduction depends on the roughness of the sea surface and the spectral frequency of the transmitted signal. For a relatively smooth surface with wave heights of approximately 0.3 meters, and a transmitted signal of 25 kHz, each reflection is estimated to cause a 3 dB loss [14]. At the operational frequencies of Seaweb (9-14 kHz) the estimated losses are less since the transmitted wavelengths are larger relative to the wavelength of the sea surface [14]. Bottom reflection losses are more difficult to predict because of the large variations in bottom type and characteristics. Additionally, the grazing angle at which the sound strikes the bottom affects the amount of loss in each interaction. However it can be expected that the losses to the bottom are on the order of 8 dB per interaction [14].

Seaweb takes advantage of these losses when determining the node-to-node ranges by using a peak-detector filter which allows the ranging algorithms to select the highest intensity multipath arrival. Since the reflected rays will generally have reduced intensity due to interactions with the boundaries, they are ignored by the filter in favor of the direct path rays.

B. GEOMETRIC ERRORS

Some of the errors that can degrade the node-to-node range data are not a result of acoustics. Rather, these errors arise from the physical geometry of the network nodes.

1. Depth Variance

The sea floor can exaggerate the measured range. Figure 13 depicts a situation where two nodes are located at different depths by virtue of the bathymetry contours. The green arrow is the horizontal range between the two nodes, and the depth difference is Δz. By the Pythagorean Theorem the measured slant range is

$$r_{measured} = \left| \sqrt{r_{actual}^2 + (\Delta z)^2} \right|. \tag{3.3}$$

This results in measured ranges that are greater than horizontal ranges between the nodes.

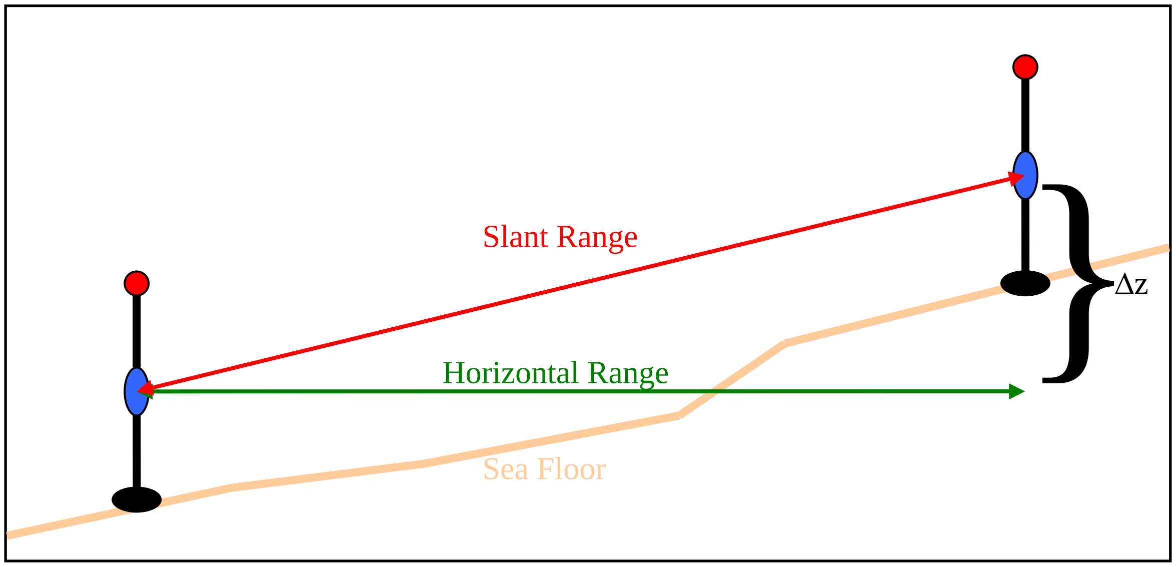


Figure 13. Range error caused by nodal depth differences.

This difference between slant range and horizontal range is neglected by the algorithm, which analyzes for a 2-D solution where all nodes are assumed to be in the same horizontal plane. This is done not only to simplify the algorithms used, but because the differences in the ranges are typically negligible, being less than 0.1% error in most cases. A 10% slope is required to produce even a 0.5% error in the measured range. Table 1 evaluates the range errors for a few geometries.

Actual Range (m)	Depth Difference (m)	% Slope	Measured Range (m)	% Error
5000	5	0.10%	5000.002	0.000%
5000	10	0.20%	5000.01	0.000%
5000	50	1.00%	5000.25	0.005%
5000	100	2.00%	5001	0.020%
2500	5	0.20%	2500.005	0.000%
2500	10	0.40%	2500.02	0.001%
2500	50	2.00%	2500.5	0.020%
2500	100	4.00%	2501.999	0.080%
500	5	1.00%	500.025	0.005%
500	10	2.00%	500.1	0.020%
500	50	10.00%	502.4938	0.499%
500	100	20.00%	509.902	1.980%

Table 1. Summary of measured range errors based on depth differences.

2. Watch Circles

The ocean is a dynamic environment where currents and tides displace the transducers even when the moorings are stationary. These rotations are commonly called watch circles and are well know to a mariner who has spent time at anchor. Based on the schematic layout of a telesonar repeater node depicted in Figure 3, the transducer is approximately 3 meters off the bottom. This results in very little variation in the maximum range difference between two nodes. If both nodes leaned 30 degrees in opposite directions, the difference in range is only 3 meters. The racom gateway buoy ranges are much more influenced by watch circles because of the longer moorings. Based on the diagram in Figure 2, the transducer could experience a watch circle of up to 78 meters from the anchor point for a 30-degree offset from vertical.

3. Geometric Dilution of Precision

The last error considered arises from the relative bearings of the nodes in the 2-D plane and the resulting angles between the nodes. Consider the case illustrated in Figure 14 where the range error (Δr) is considered to be the same for each pair of nodes. When the two reference nodes are located near each other (Figure 14 a), the triangularized position of the unknown node has a large uncertainty region referred to as geometric dilution of precision (GDOP). As the reference nodes are moved further apart (Figure 14 b) and the angle θ_{tr} between them increases, the GDOP is reduced until it reaches a minimum when θ_{tr} is 90 degrees. Since the error is a function of the geometric bearings of the nodes, careful planning of the node locations can minimize GDOP.

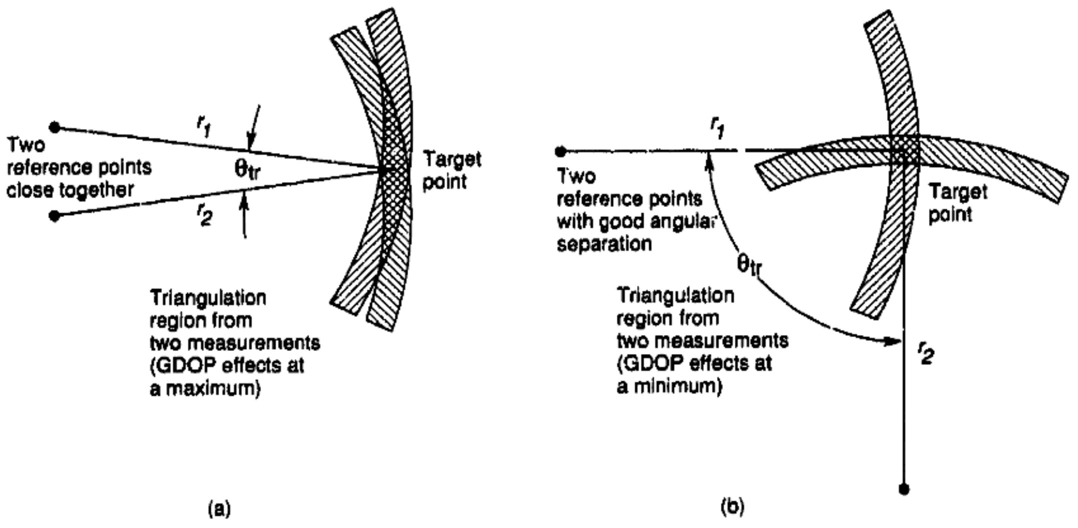

Two reference points close together
r_1
θ_{tr}
r_2
Target point
Triangulation region from two measurements (GDOP effects at a maximum)
(a)
Two reference points with good angular separation
r_1
θ_{tr}
r_2
Target point
Triangulation region from two measurements (GDOP effects at a minimum)
(b)

Figure 14. Geometric Dilution of Precision (from [15]).

C. EXPECTED ACCURACY

While all of the range errors are present in every scenario, the magnitude of their contributions depends on the location, layout, and environmental conditions. Nevertheless, some generalizations can be made about the localization error budget. The largest sources of error are the ray path assumption - which results in an overestimate of the range - and the watch circle of the racom buoy - which can cause an overestimate or underestimate of the range. The assumption that sound speed is c_o = 1500 m/s can result in significant range errors in either direction, and the effects of varying depth are typically small but tend to overestimate the range. Qualitatively, the sum of all of these errors results in an overestimation of range.

To get a feeling for the range error budget, consider a set of nodes that are actually 4,000 meters apart with a 2% slope between the nodes. Additionally assume a sound-speed profile that results in an average sound velocity of 1480 m/s. Since the racom buoy produces the largest watch circle errors, choose this as one of the two nodes. Summing all the range errors, the measured range would be reported as 4278.5 meters which is a 7% error. A breakdown of the contribution of each error type is contained in Table 2.

	Range Error (m)	% Error
Sound Speed	54	1.34
Ray Path	120	3.00
Depth Variance	1	0.02
Watch Circle (racom)	103	2.58
Watch Circle (telesonar)	1.5	0.04
Total Error	278.5	7

Table 2. Range Error Budget for two nodes 4000 meters apart with a 2% slope.

IV. LOCALIZATION METHODS

The localization of nodes in a Seaweb network is based on the range-fixing concept of using intersecting circles centered on known locations. For the purpose of this thesis, an ‘unknown’ node is one that has not yet been localized and a ‘known’ node is one that has been localized on the *x-y* plane either by a priori information, or by a range-fixing algorithm. Localization by intersecting circles works well when no node positional errors or range errors are present, but difficulties can arise when errors are introduced. Previous work by [2] and [3] attempted to mitigate these errors by using weighted averages and center of mass method, respectively. Both were tested in [4] and compared with the difference linearization method outlined by [16]. The difference linear method was found to be superior in estimating the position of unknown nodes.

Depending on the number of range measurements between an unknown node and neighboring known nodes, three possibilities exist for the type of solution that can be found for the position of the unknown node. These are one range, which results in no solutions, two ranges, which result in ambiguous solutions, and three or more ranges which result in an exact solution. These cases are discussed in the following sections. Ranges from one unknown to other unknown nodes result in no additional useful information when attempting to localize the unknown node.

A. NO SOLUTIONS

If the unknown node has only one range to a known node, there are infinitely many possible locations for the unknown node. The solution set is a circle centered at the location of the known node with a radius equal to the range from the known node to the unknown node. For this reason, unknown nodes with only a single range are not evaluated and are passed over until the algorithm has localized a second node for which the unknown node has a range, resulting in the case described in the next section.

B. AMBIGUOUS SOLUTIONS

When the unknown node has ranges to two neighboring known nodes, there are two ambiguous solutions that exist where the two range circles intersect. One will be above a line that intersects the two known nodes, and the other is a reflection below the line. This is depicted in Figure 15.

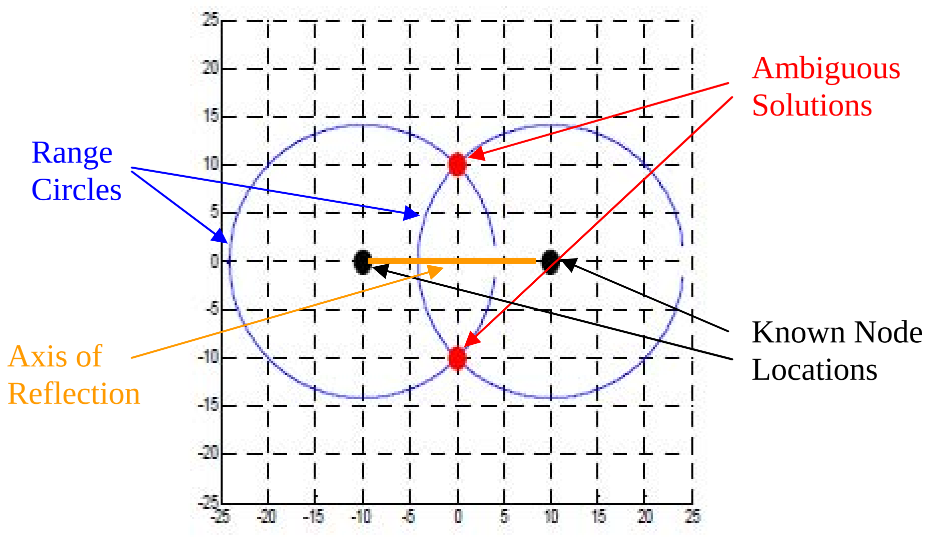


Figure 15. Two-node, two-range solution set (after [3]).

Determination of the locations of the two ambiguous solutions is made using the law of cosines. Figure 16 demonstrates the locations of the two known nodes, *i* and *j*, and the two red triangles symbolize the unknown node's ambiguous solutions. The node-to-node ranges from the known nodes to the ambiguous solutions are represented by r_i and r_j, and r_{ij} is the range along the axis of reflection between the two known nodes. The angle from the axis of reflection to the bearing of the unknown node is determined using the law of cosines

$$\theta = \cos^{-1}\left(\frac{r_i^2 + r_{ij}^2 - r_j^2}{2r_i r_{ij}}\right). \tag{4.1}$$

The bearing to the ambiguous solution is simply the negative of that angle. Once θ is determined, the *x* and *y* coordinates of the unknown node are given by

$$\begin{aligned} x &= r_i \cos\theta \\ y &= r_i \sin\theta \end{aligned} \quad . \tag{4.2}$$

From Figure 15 it is noted that the *x* coordinates are the same for both ambiguous solutions and the *y* coordinates are reflections about the *x*-axis.

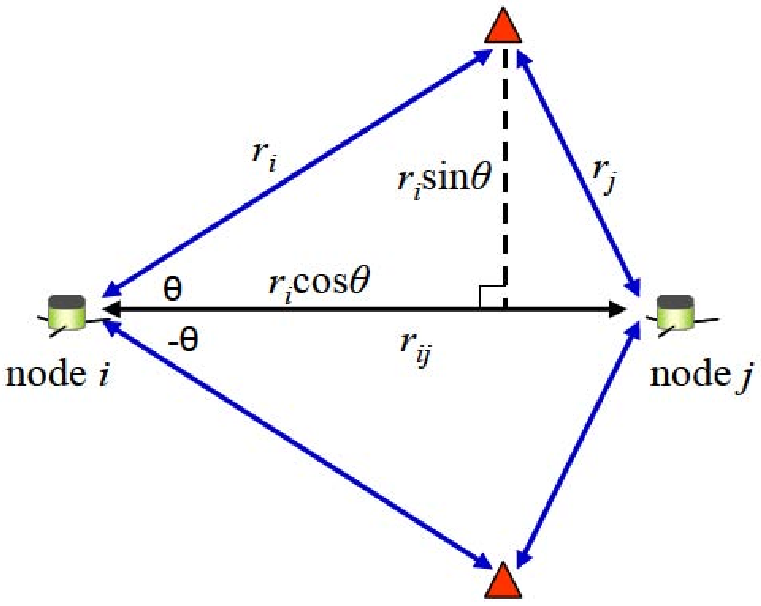


Figure 16. Law of cosines used to determine the angle θ to the unknown node location from node *i*, a known node (from [2]).

C. DIFFERENCE LINEARIZATION METHOD

When an unknown node has ranges to three known nodes, a single unambiguous solution to the location of the unknown node exists where the three range circles intersect, as seen in Figure 17. This trilateration is the basis for radar navigation and the Global Positioning System.

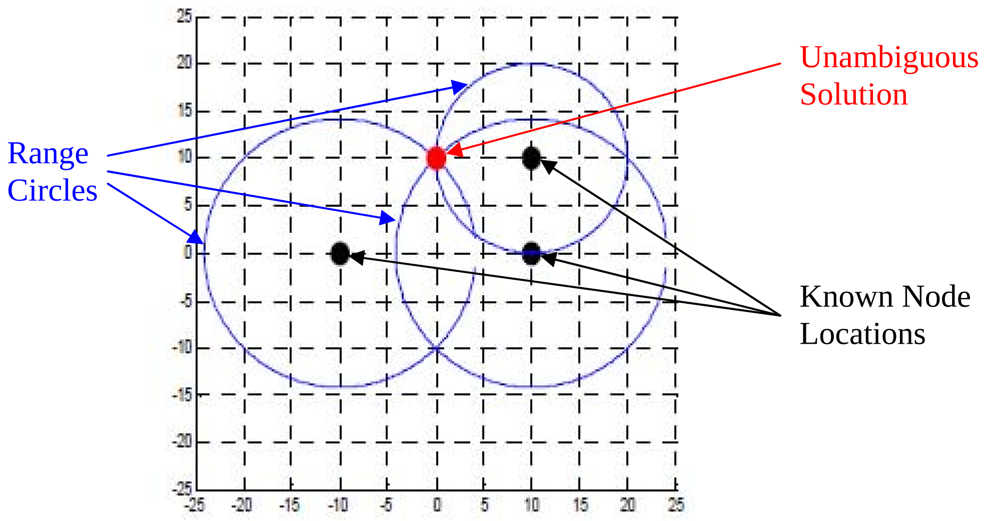


Figure 17. Three-node, three-range solution (after [3]).

The difference linearization method is based on the fact that in an *n*-dimensional coordinate system, the location of an object can be found if ranges are known to *n*+1 fixed locations. In a 2-D system, the distance r_i between the *i*th point at a known position and the object located at (*x*, *y*) is given by the Pythagorean theorem: $r_i^2 = (x - x_i)^2 + (y - y_i)^2$. Since this is a 2-D system, the distances to three known locations may be written as a system of equations of the form

$$\begin{bmatrix} r_1^2 \\ r_2^2 \\ r_3^2 \end{bmatrix} = \begin{bmatrix} (x-x_1)^2 + (y-y_1)^2 \\ (x-x_2)^2 + (y-y_2)^2 \\ (x-x_3)^2 + (y-y_3)^2 \end{bmatrix}. \tag{4.3}$$

The difference linearization method described in [16] is applied to this system of equations. This method first eliminates the square terms and reforms the equality so that it is the sum of a pair of simultaneous equations in *x* and *y*,

$$\begin{bmatrix} (r_1^2 - r_2^2) \\ (r_2^2 - r_3^2) \end{bmatrix} = 2\begin{bmatrix} (x_2 - x_1)(y_2 - y_1) \\ (x_3 - x_2)(y_3 - y_2) \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} (x_1^2 - x_2^2) + (y_1^2 - y_2^2) \\ (x_2^2 - x_3^2) + (y_2^2 - y_3^2) \end{bmatrix}. \tag{4.4}$$

Solving for the location of the object (*x*, *y*) results in

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{2}\begin{bmatrix} (x_2 - x_1)(y_2 - y_1) \\ (x_3 - x_2)(y_3 - y_2) \end{bmatrix}^{-1}\begin{bmatrix} (r_1^2 - r_2^2) + (x_1^2 - x_2^2) + (y_1^2 - y_2^2) \\ (r_2^2 - r_3^2) + (x_2^2 - x_3^2) + (y_2^2 - y_3^2) \end{bmatrix}. \tag{4.5}$$

It is possible for an unknown node to have ranges to more than three known nodes causing Equation (4.3) to be over-determined. Two possibilities then exist for estimating the location of the node. Either the over-determined system of equations can be solved simultaneously using a least-squares procedure, or the ranges can be selected in combinations of three and the position estimated as a weighted average. When the range data contains errors, the latter method was found to be superior by [4] due to its ability to successfully mitigate range errors. For this reason using combinations of ranges and averaging the solutions is implemented in this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

V. ALGORITHM IMPLEMENTATION

The algorithm developed in this thesis is implemented in the MATLAB programming language and consists of four basic stages: operator input, data input, removal of outliers from the data, and determining the locations of the nodes. Each stage is written as a separate function that is called by a master program. The MATLAB code is listed in Appendix A.

A. OPERATOR INPUT

The first stage requires the operator to provide the parameters associated with the Seaweb network to be analyzed. The operator identifies the address of each node in the network, and specifies the 'master' node and the 'on_X' node. The 'master' node is usually the gateway buoy that supports interface between the underwater network and the operator. A useful attribute of the gateway buoy is its on-board GPS receiver. The 'on_X' node is the only other node in the network that must be specially designated. It must be close enough to the 'master' node to ensure that a range measurement is available and the operator must be able to specify what the true bearing is from the 'master' node to the 'on_X' node. This permits the operator to orient the solution produced by the algorithm to a geo-referenced coordinate system, such as latitude-longitude.

The next thing the operator specifies are the names of the text files that contain the node-to-node range data. The format of the text data files is shown in Table 3, below.

3	30	75.1	10	120.6	20	294.3	11	412.8
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
20	11	170.8	10	182.0	30	237.3	3	300.1
30	10	70.6	3	78.5	20	235.4	11	396.6

Table 3. Example of the node-to-node range data format used by the algorithm. The red column is the node address that initiated the communication, the blue column is the node address that responded, and the black column is the calculated range between the two nodes in meters. Zeros indicate that the initiating node did not receive a response.

B. STACK BUILD

The Stack_Build.m function takes the input data and sorts it into a single 3-D matrix, referred to as a range stack, in a form suited for analysis. The range stack consists of the same number of layers as the number of files loaded by the operator. Each layer is a square matrix where the node addresses comprise the first row and first column of the matrix and serve as headers for indexing the range data. For convenience, the 'master' node is assigned to the second row/column and the 'on_X' node is assigned to the third row/column.

Each node-to-node range data file is loaded and its contents are organized as a layer in the stack with each range datum stored in the row of the node that initiated the communication and in the column of the corresponding node. Once a layer is completed, it is placed on the stack and the next data file is loaded. A graphical representation of the range stack appears in Figure 18, below.

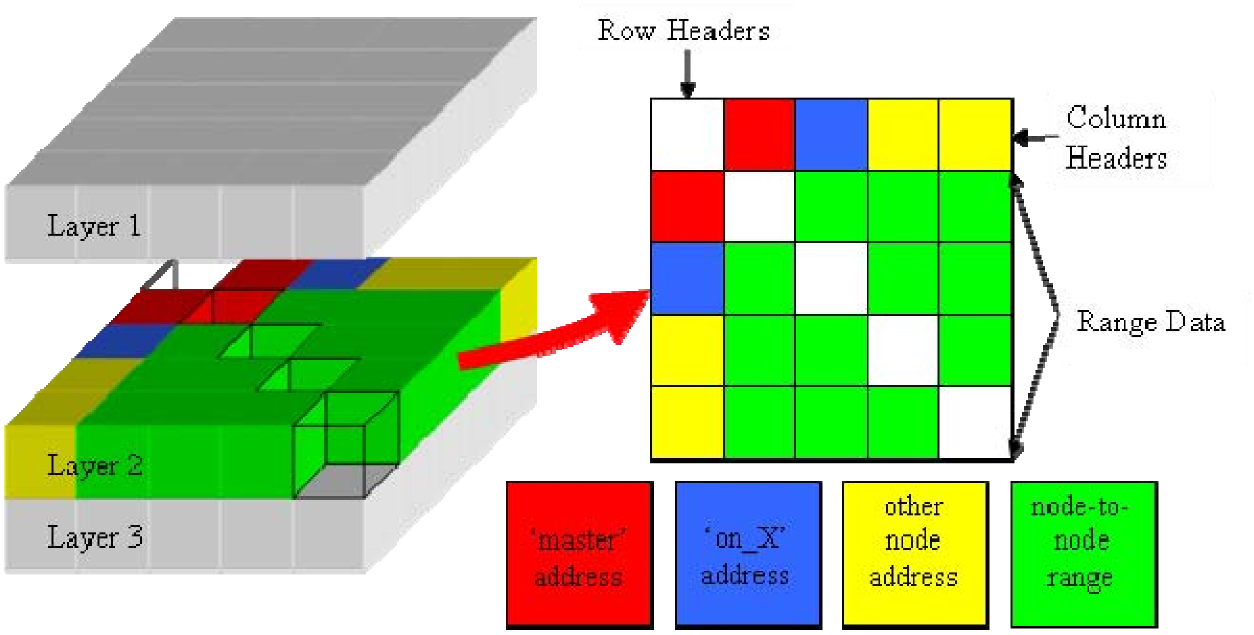


Figure 18. The range stack is constructed from data files.

C. STACK SIFT

The range stack is passed to the function Stack_Sift.m, where it is distilled to determine the mean range between each pair of nodes. Each node pair is statistically evaluated and the resulting mean ranges are stored in a single 2-D array.

The first step is for all the ranges for a particular node pair to be loaded into a single vector. This is done by "looking down the stack" for the particular node pair for all the range measurements made during the discovery process (e.g., node 3 to node 7). Simultaneously the reciprocal combination is found by looking "up the stack" (e.g., node 7 to 3). This process is depicted in Figure 19.

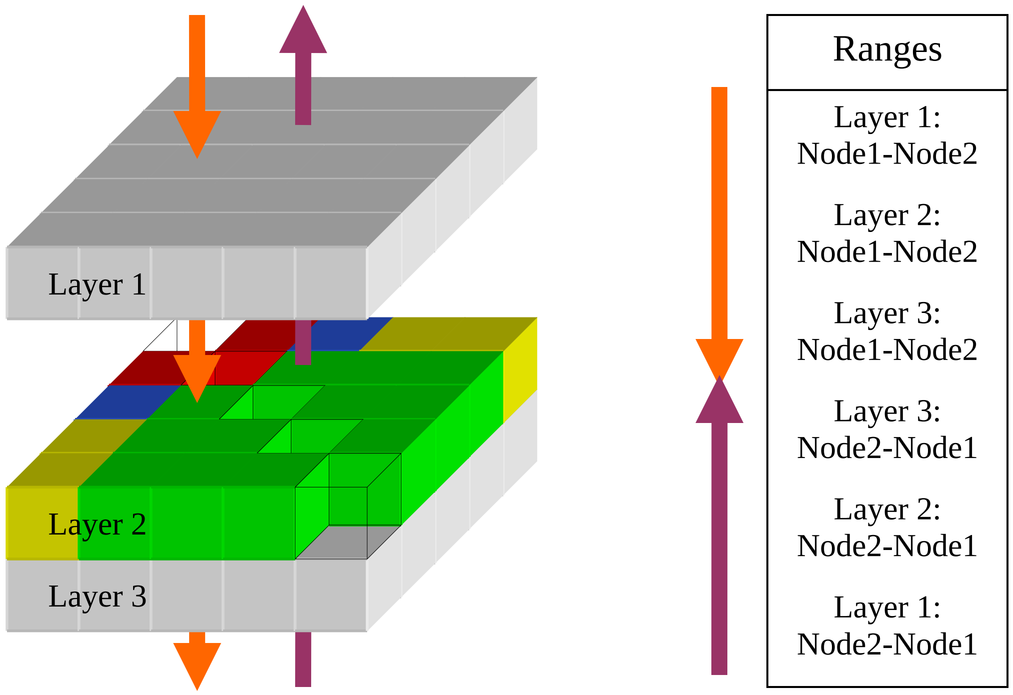


Figure 19. Example of building the vector of range data for a node pair.

The vector is statistically analyzed in an iterative manner to derive a good range between the selected nodes by progressively eliminating outlier ranges that fall outside a ten percent confidence interval for the data. Initially the mean and ten-percent confidence interval of the data ranges are calculated and data that fall outside the confidence interval are flagged. The flagged datum falling furthest from the mean is eliminated. The mean and confidence interval are then recalculated and the process is repeated until no data fall outside the confidence interval. The final calculated range estimate for the node pair is then stored in the appropriate element of the 2-D array. Once all node pairs have been analyzed, the resulting output is a 2-D square array that is symmetric about the main diagonal, with node pairs that lack a range estimate denoted by 'NaN' ("Not a Number").

NaN	'master' Node	'on_X' Node	Node 3	Node 4
'master' Node	NaN	Range: m -> X	Range: m -> 3	Range: m -> 4
'on_X' Node	Range: m -> X	NaN	Range: X -> 3	Range: X -> 4
Node 3	Range: m -> 3	Range: X -> 3	NaN	Range: 3 -> 4
Node 4	Range: m -> 4	Range: X -> 4	Range: 3 -> 4	NaN

Table 4. Example output range array from Stack_Sift.m.

D. SOLVER

Now that the range data are filtered, averaged, and organized, the Solver.m function estimates the locations of the nodes. The first step is to establish a frame of reference based on the known locations of ‘master’ and ‘on_X’ nodes. The master node location is set to be at the origin $(0,0)$ of a horizontal Cartesian grid. The positive *x*-axis is forced to intersect the location of the ‘on_X’ node, such that the coordinate of this node is $(r_{master \to on-X}, 0)$.

Once a node is localized within the grid, it is considered to be a known node. With these known nodes in place, the algorithm evaluates each remaining node to determine how many ranges it has to known nodes in the grid. Three distinct possibilities exist: one range, two ranges, or three or more ranges. If only one range exists to a known node no attempt is made to localize that node since the solution set is a circle centered at the known node’s location with a radius equal to the range.

If a node has ranges to only two other known nodes, there are two ambiguous solutions to the location of that node. Since no solutions can be eliminated without knowing some geographical constraints, all possible solutions are tracked. This means that there are potentially $2^{(\text{\# of nodes}-2)}$ different possible solutions that must be stored. To account for all these possibilities, a binary system is employed that tracks each time a node has ranges to only two nodes. For example, in a five-node network there are eight possible solutions. The first two nodes, the ‘master’ and ‘on_X’ nodes are in fixed locations, and the third node has ambiguous solutions that are labeled 00000 and 00100 respectively. If the fourth node also has only two ranges to nodes with known locations, there are four possible solutions: 00000, 00100, 00010, and 00110. (These solutions are alternately identified by their base-ten values 0, 4, 2, and 3). When a previously unknown node becomes known, the program also sets a flag to cause it to cycle through all the nodes again since a node that had previously been bypassed may now have sufficient ranges to known nodes to make its localization possible.

When a node has ranges to three known nodes, the program then finds a single solution using Equation (4.5). If this is the first time a solution has been determined for this node, the program sets the flag to cause the program to loop through all nodes again and also has that node evaluated in all possible ambiguous solution sets. If it not the node’s first solution, the location determined by this particular iteration of the algorithm is checked against the previous solution. If either the *x*-coordinate or the *y*-coordinate are greater than some range difference threshold from the previous solution, the new solution is considered improved and the program sets the flag to have every node in every ambiguous solution set evaluated again.

This process continues until the all nodes have been located within the specified tolerances of the previous solutions. The output of the program is a list of all the ambiguous solutions, i.e., 00000, 00100, 00110, etc…, as well the locations of each node for each ambiguous solution set.

E. EXAMPLE OF A FIVE-NODE LAYOUT

This section provides an example of the algorithm process for a simple five-node layout for illustration purposes. The nodes are generated randomly on a 5000-meter by 5000-meter grid, the ‘master’ node is node 4, and the ‘on_X’ node is node 2. The true node locations and the actual node-to-node ranges are given in Table 5. These true node locations are plotted in Figure 20. Figure 21 is a plot of the same five nodes with the frame of reference translated such that the origin is at the ‘master’ node and the positive *x*-axis intersects the ‘on_X’ node.

Node Locations (meters)		
Node	*x*-position	*y*-position
4	1393	2735
2	635	4567
3	3162	488
1	4074	4529
5	4788	4825

Node-to-Node Ranges (meters)					
NaN	1	2	3	4	5
1	NaN	3439	4143	3226	773
2	3439	NaN	4798	1983	4161
3	4143	4798	NaN	2860	4632
4	3226	1983	2860	NaN	3987
5	773	4161	4632	3987	NaN

Table 5. Example node locations and ranges.

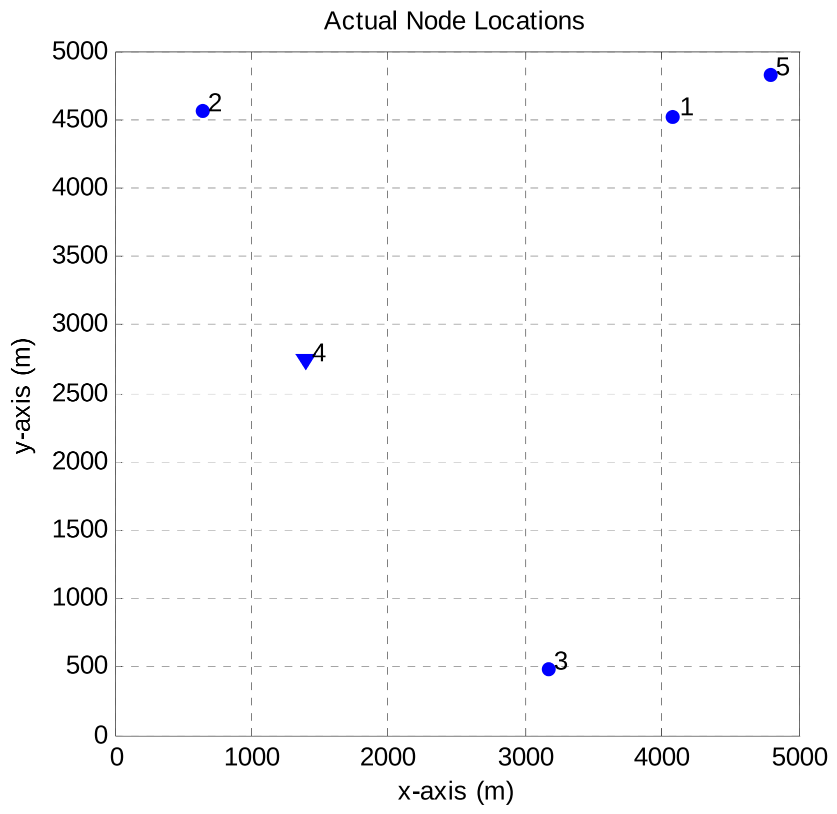


Figure 20. Locations of example nodes.

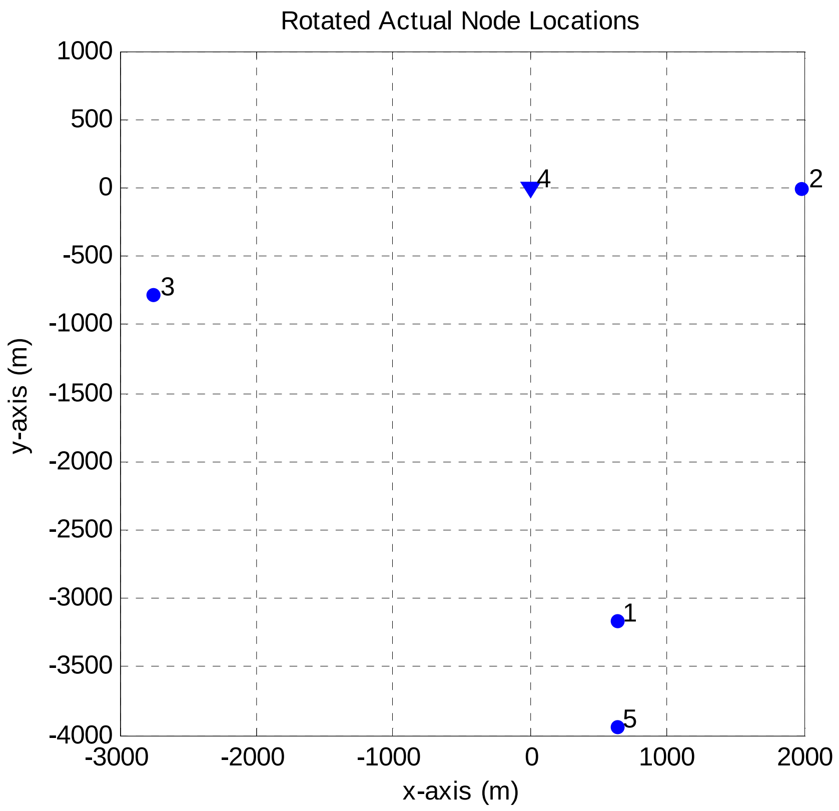


Figure 21. Example nodes rotated and translated to place 'master' at origin and 'on_X' on the positive *x*-axis.

For realism, associated with limitations of acoustic communications, ranges in excess of 4000 meters are removed, resulting in the range matrix shown in Table 6. This is input into the Solver.m function.

Node-to-Node Ranges					
NaN	4	2	3	1	5
4	NaN	1983	2860	3226	3987
2	1983	NaN	NaN	3439	NaN
3	2860	NaN	NaN	NaN	NaN
1	3226	3439	NaN	NaN	773
5	3987	NaN	NaN	773	NaN

Table 6. Node-to-node range matrix after 4000-meter maximum range cutoff is applied.

Solver.m produces the following ambiguous solutions: 0, 1, 2, and 3, all of which are depicted in Figure 22. Comparing these solutions to Figure 21, it is apparent that ambiguous solutions 0 and 1 are on the wrong half-plane based on nodes 1 and 5 being below the *x*-axis in Figure 21 and nodes 1 and 5 being above the *x*-axis in solutions 0 and 1. Solution 2 is eliminated due to nodes 1 and 5 being in nearly a straight vertical line in Figure 21, while solution 2 has these nodes canted. Thus, solution 3 is the most accurate solution. Also of note is that node 3 is not featured in any solution set because it has only a single range to other nodes after the 4000-meter range cutoff is applied.

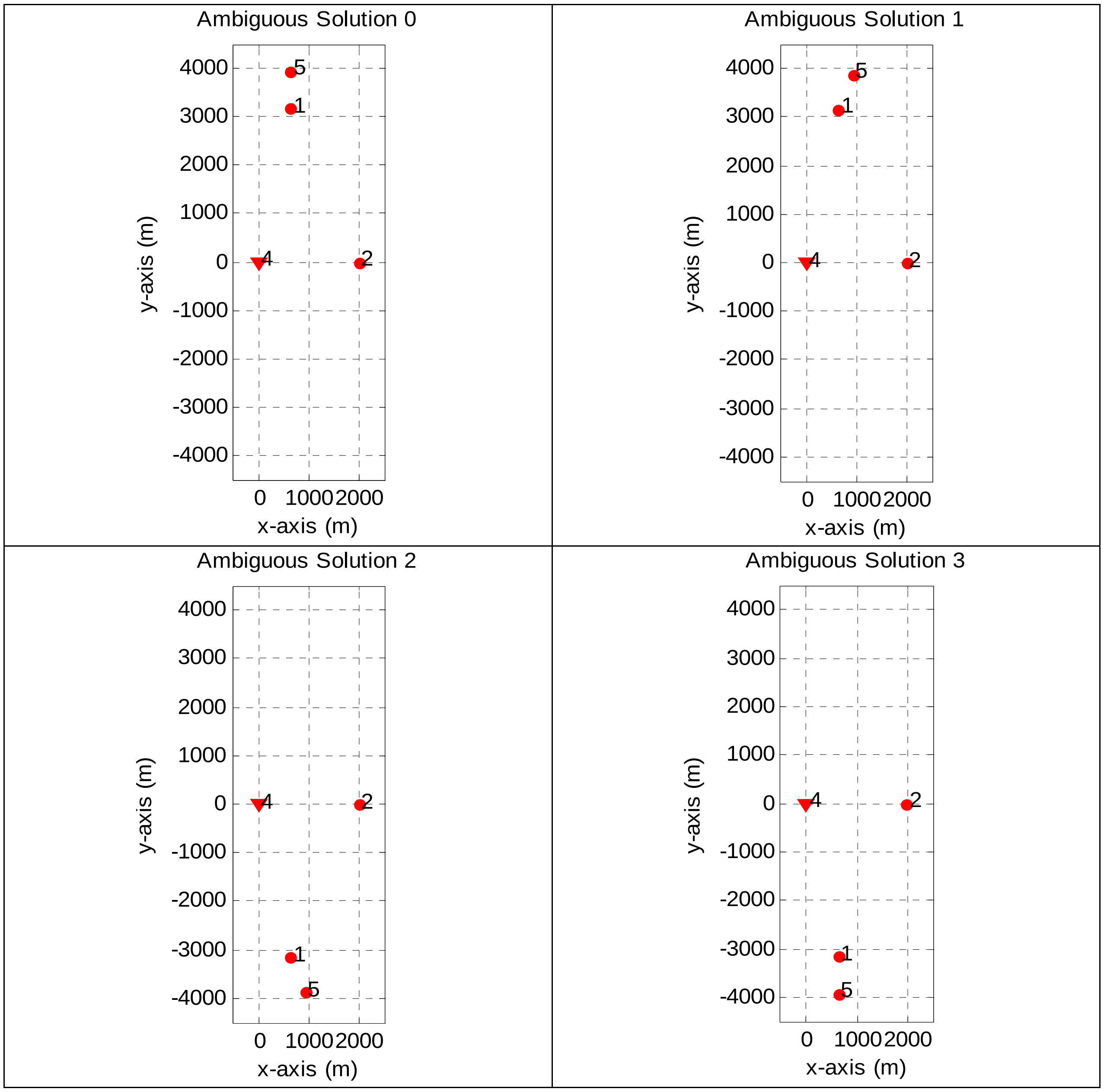


Figure 22. Plotted results of all ambiguous solutions of example nodes.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. SYNTHETIC DATA ANALYSIS

Synthetic data for testing the algorithm are created in MATLAB using the built-in random number generator functions. For the node locations, the function `unidrnd` gives a uniformly distributed random number for both the *x* position and the *y* position. Error-free ranges are calculated for the node pairs. Simulated range error is introduced by multiplying the error-free range by a normal distribution, and then adding five percent of the actual range multiplied by a chi-squared distribution,

$$r_{reported} = Normdist(\mu, \sigma) \times r_{actual} + 0.05 \times r_{actual} \times \chi^2(\nu) \,. \tag{6.1}$$

The normal distribution introduces some variability to the reported ranges to account for sound speed assumption errors which tend to be small and proportional to the range between the nodes. The mean, μ, is set to one, and the standard deviation, σ, is 0.1 to ensure that most reported ranges will be near the actual range. The chi-squared distribution is chosen as the model for the additive factor to account for refracted ray path errors and depth variance errors which tend to be larger, but also proportional to the range between the nodes. The parameter ν is set to one for this evaluation, resulting in an expected value of one, with a standard deviation of two. When this is multiplied by 5% of the error-free range, it results in an approximate 5% overestimation of the range. This method typically yields a reported range greater than the actual range. This reflects the fact that most range errors result in an overestimation of the range as established in the error analysis performed in Chapter III. Watch circle errors are not addressed since they normally only affect the racom node and will typically average out over the course of a network's operations.

For analysis of the algorithm performance, fifteen nodes are randomly placed in an eight-kilometer by six-kilometer region, since this is the approximate area of the St. Margaret's Bay sea trial. For each simulation, the node-to-node range calculations are performed fifty times with different realizations of the range errors. Each set of ranges is entered as a separate layer in the range stack. To match realistic acoustic communication range limitations, ranges exceeding 4000 meters are omitted. For consistency across all simulations, node 1 is set as the 'master' node and node 2 is set as the 'on_X' node. The algorithm is allowed an arbitrary maximum of 50 iterations to achieve convergence of the node coordinates. For the simulation cases considered in this thesis, 50 iterations is sufficient for convergence of the node positions. Increasing the number of allowed iterations was found to not significantly increase the accuracy of the solutions.

A. OUTLIER RANGE REJECTION

Simulation testing confirms that the Stack_Sift.m program rejects ranges that fall outside the confidence interval and estimates a good range between nodes. For the analysis in this section, the 4000-meter range cutoff is removed to maximize the number of ranges evaluated in each pass. The analysis is performed for 100 realizations of the fifteen-node network, resulting in 19,600 ranges being evaluated. The percent error between the actual ranges and the ranges with offsets is recorded, and then the ranges are analyzed by the Stack_Sift.m function. The percent error between the output of the function and the actual ranges are then computed.

The mean percent range error for the ranges not run through Stack_Sift.m is 4.7%, and the largest percent error is 5.0%. After the ranges are processed by Stack_Sift.m, the mean error is reduced to 3.7%, and the single largest error is 4.3%. This filtering gains over a 1% increase in the accuracy of the range data, which is substantial when localizing nodes. In absolute terms, the accuracy of a 4000-meter range estimate is improved by 40 m. The results are summarized in Table 7.

	Mean % Error	Max % Error	Min %Error
Pre Stack_Sift.m	4.7%	5.0%	4.3%
Post Stack_Sift.m	3.7%	4.3%	3.0%

Table 7. Percent Error improvement gained by Stack_Sift.m.

B. ERROR-FREE RANGES

Error-free testing is performed to verify the ability of the algorithm to correctly determine the positions of all nodes with perfect range data. Several statistics are computed for each network realization. The number of iterations required by the algorithm to position all nodes within 25 meters accuracy is recorded because this is indicative of the processing time required. The program also tracks how many of the fifteen nodes have been localized to ascertain how many of the nodes are beyond the maximum cutoff range to more than one node. Thirdly, the algorithm counts the total number of ambiguous solutions caused by an unknown node having ranges to just two known nodes. This count is important because it indicates how many ambiguous solutions the operator must evaluate for redundant or impossible solutions, such as those that result in a node being located on land when translated to a navigation chart. Lastly, the true positions of all the nodes are translated and rotated to place the ‘master’ node at the origin and the ‘on_X’ node on the *x*-axis. The positions of each node in each ambiguous solution are then compared to these true positions, and the ambiguous solution that has the lowest total range error is recorded. This total error is divided by the number of nodes that were localized for that particular node set, to give the mean error per node for that set.

One-hundred realizations are simulated to yield an adequate sample size. In four of these 100 sets, the algorithm is unable to determine the location of a third node due to no nodes being within the maximum cutoff range (4000 meters) of both the ‘master’ node and ‘on_X’ node. These four realizations are excluded from analysis since they would skew the statistics of interest.

The first statistics analyzed are how many of the fifteen nodes are able to be localized and how many iterations of the program are required to converge all nodes to within 25 meters of its previous solution. The mean number of nodes localized is 14.8 of the possible 15. The mean number of iterations is 19.1, but this result is bi-modally skewed by the large number of times the program required less than five iterations and the number of times it required all 50 allowed iterations. More indicative statistics are the median number of iterations (6.5) and the mode (3). Figures 23 and 24 provide histograms of the results as well as the statistical values of interest.

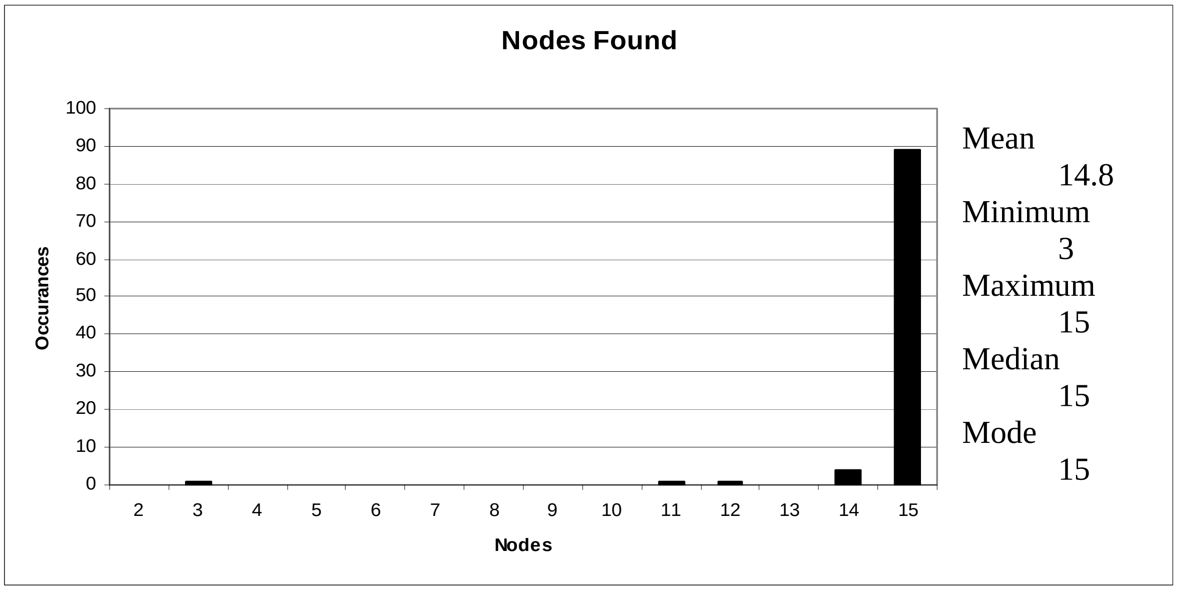


Figure 23. Total number of nodes localized for 100 realizations of 15-node networks in an 8-km by 6-km area, using error-free ranges.

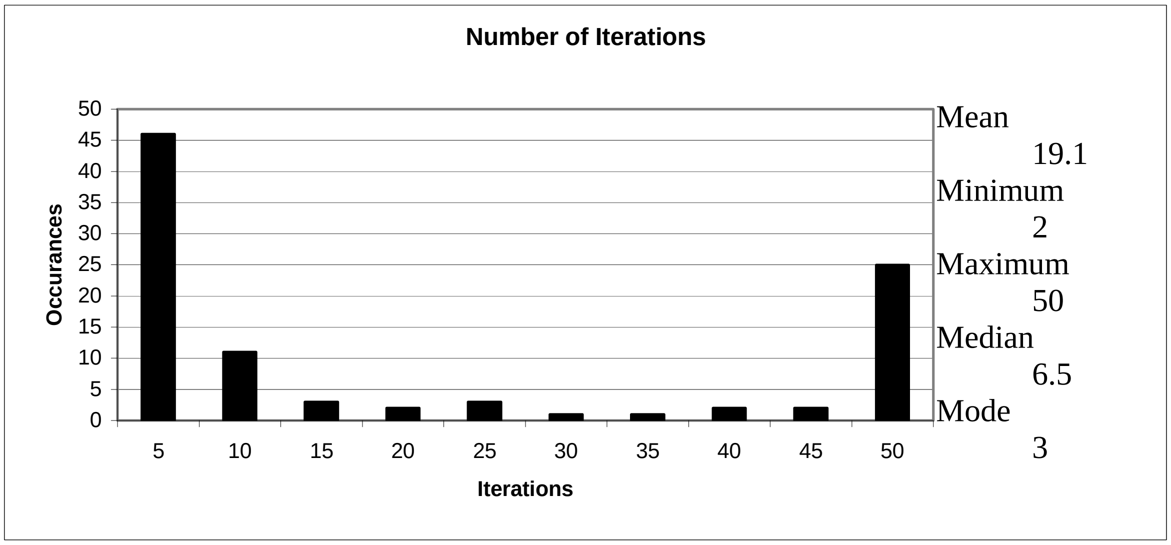


Figure 24. Total number of iterations for 100 realizations of 15-node networks in an 8-km by 6-km area, using error-free ranges.

The number of ambiguous solutions is also evaluated. The mean number of ambiguous solutions is 11.1, and the maximum is 64. For a fifteen-node network the minimum number of solutions is 2, with a theoretical maximum of 8192 solutions. The results are summarized in Figures 25.

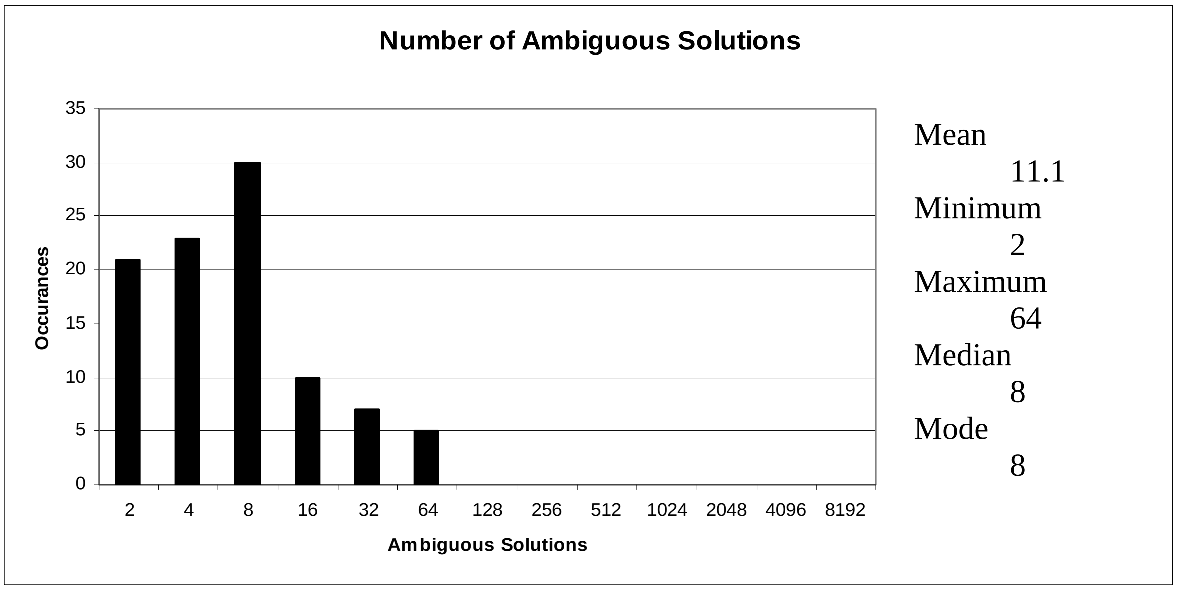


Figure 25. Total number of ambiguous solutions for 100 realizations of 15-node networks in an 8-km by 6-km area, using error-free ranges.

The most telling statistic is how well the algorithm is able to localize the nodes when compared to the actual positions. This is done by finding the mean error for each node for a particular solution set. The mean nodal localization error is 5.8 meters with a maximum of 242 meters. The minimum, mode and median of the localization errors are all less than one. Figure 26 presents the data as a histogram. It is unclear at this point why range errors are incurred when the known ranges between nodes are exact. It is presumed that the maximum number of allowed iterations and acceptance criteria for the algorithm to not flag an ambiguous solution for another iteration play major roles.

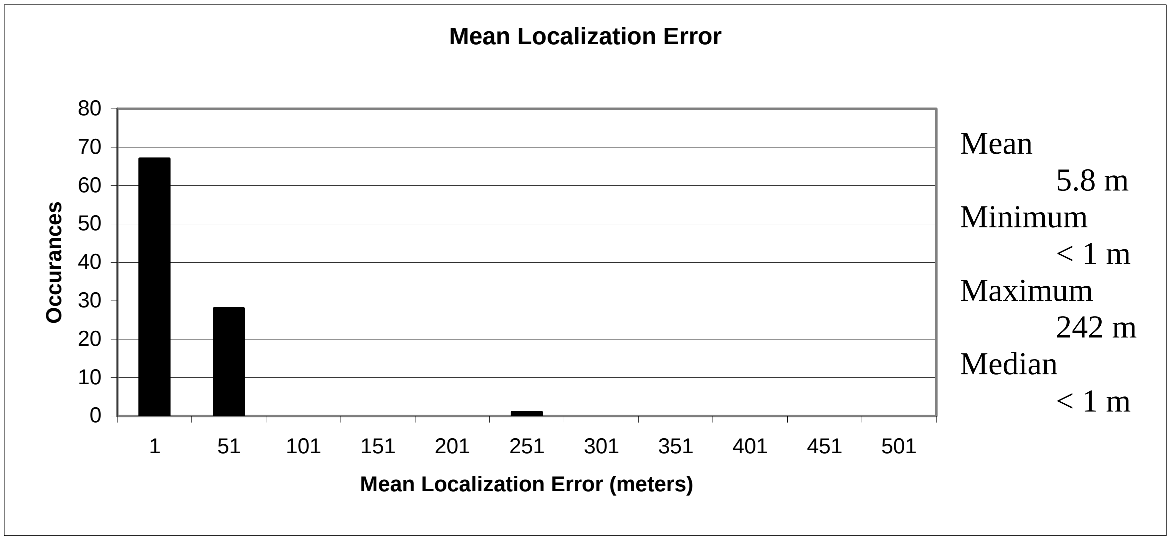


Figure 26. Mean localization error for 100 realizations of 15-node networks in an 8-km by 6-km area, using error-free ranges.

C. ERROR-INDUCED RANGES

The same analyses that are performed on the error-free realizations are repeated for ranges that include errors introduced with Equation (6.1). The results of the algorithm show that in 35 of the node sets no solutions are achieved due to no nodes being within the maximum cutoff range of both the 'master' and 'on_X' nodes. The greater number of node sets that the algorithm was unable to localize is partly due to the fact that the range errors introduced resulted in a greater number of nodes that had ranges above the 4000-meter cutoff to the 'master' and 'on_X' nodes.

As expected, the performance is poorer than in the error-free cases. The mean number of nodes localized declines to 8.9 nodes per set. Also the number of iterations required increases to nearly the maximum in almost every simulation, rising to a mean of 47.5 passes. The data are presented in Figures 27 and 28.

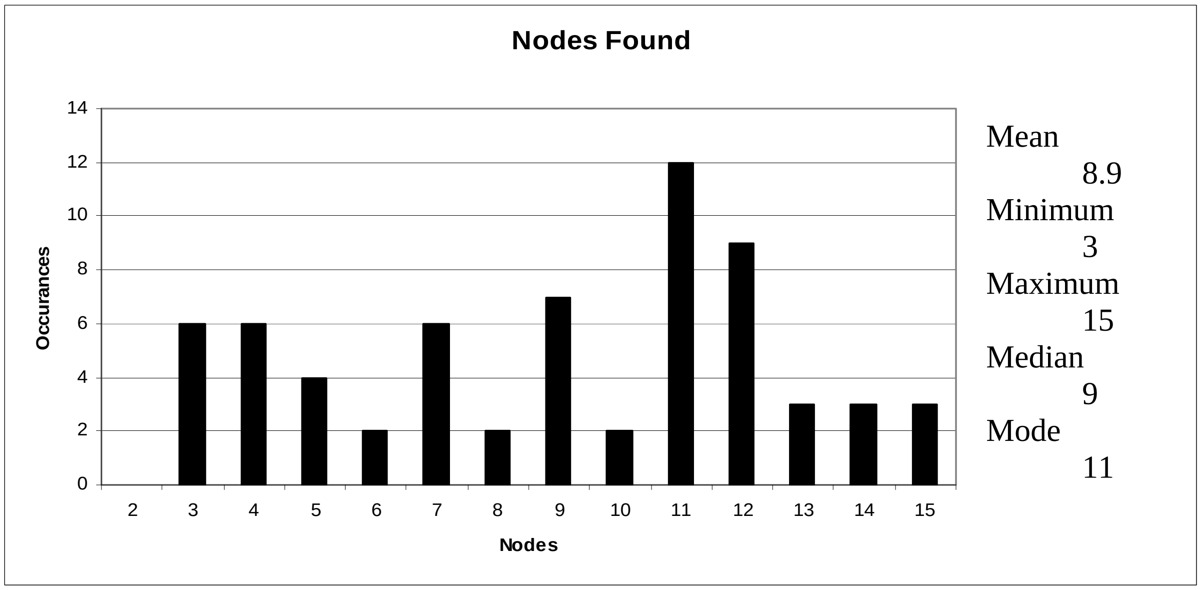


Figure 27. Total number of nodes localized for 100 realizations of 15-node networks in an 8-km by 6-km area, with range errors.

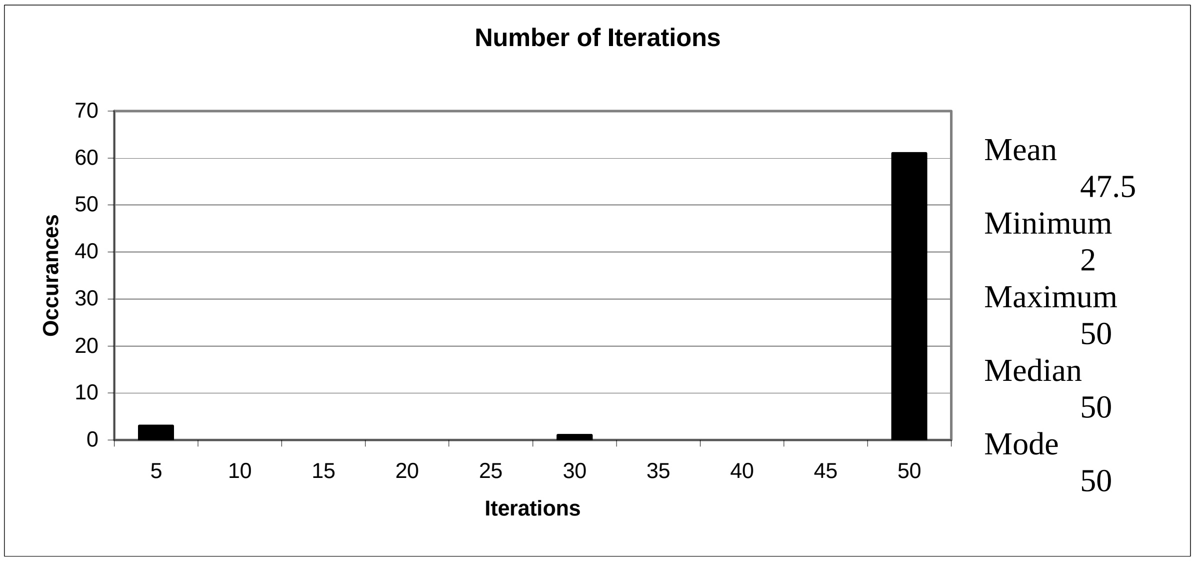


Figure 28. Total number of iterations for 100 realizations of 15-node networks in an 8-km by 6-km area, with range errors.

The number of ambiguous solutions also increases over the error-free case. The mean number of ambiguous solutions increases to 226.2 with a maximum of 4096 ambiguous solutions. Figures 29 shows these results.

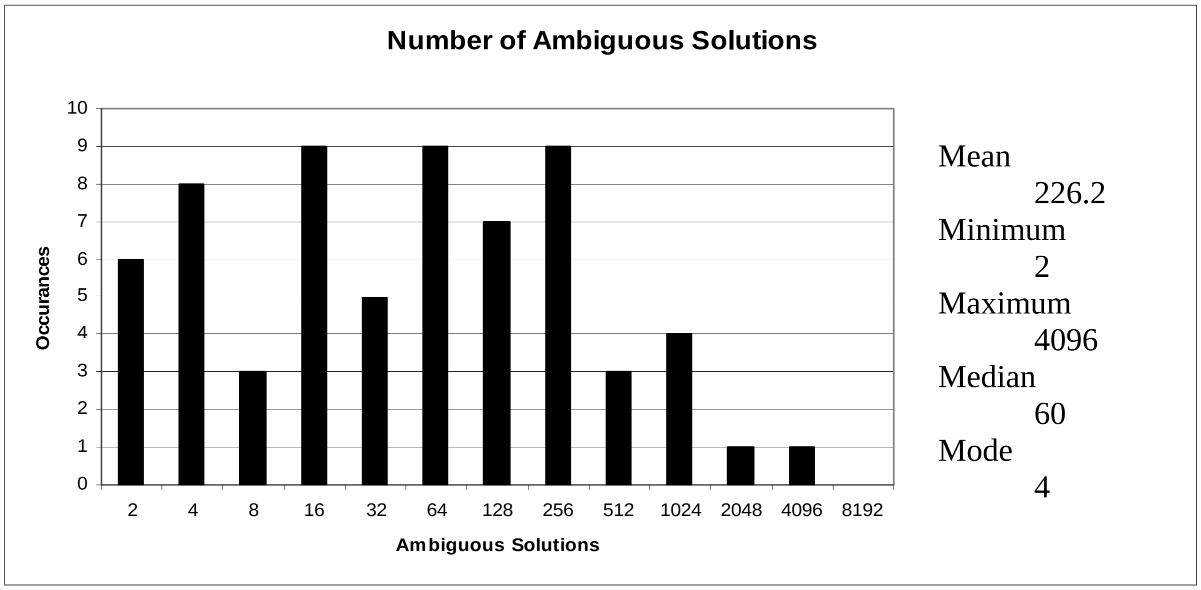


Figure 29. Total number of ambiguous solutions for 100 realizations of 15-node networks in an 8-km by 6-km area, with range errors.

Obviously, the mean best localization per node also increases. The mean value is 511 meters with a maximum of 3811.7 meters. However the median is 312 meters, meaning that the localization error tends to be less than a quarter of a nautical mile. Figure 30 summarizes the results.

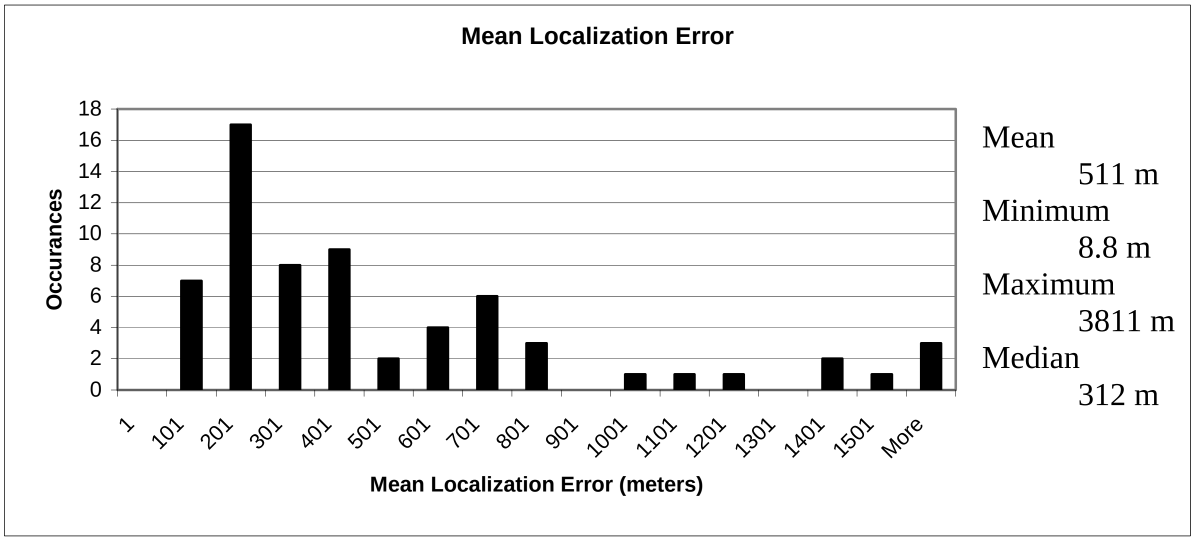


Figure 30. Mean localization error for 100 realizations of 15-node networks in an 8-km by 6-km area, with range errors.

VII. ST. MARGARET'S BAY EXPERIMENT

In June of 2008, Naval Postgraduate School conducted a test of the firmware upgrade of the ad hoc discovery scheme developed in [1] at St. Margaret's Bay, Halifax, Nova Scotia, Canada as part of the 2008 TTCP Unet sea trials.

A. SETUP

Nineteen nodes were deployed in this experiment, including a single racom buoy that served as the master node and which was located near the center of the Seaweb network. Figure 31 shows the physical components of the Seaweb network. Table 8 tabulates the recorded deployment location of each node as determined by GPS fixes. Figure 4 shows a chart of St. Margaret's Bay and the location of each node.

The water where the nodes were deployed varies from 30 to 70 meters deep, with the bottom type characterized as sand or gravel. Wind speed was typically less than 8m/s as observed at CFAV Quest, and shipping traffic was observed to be light.

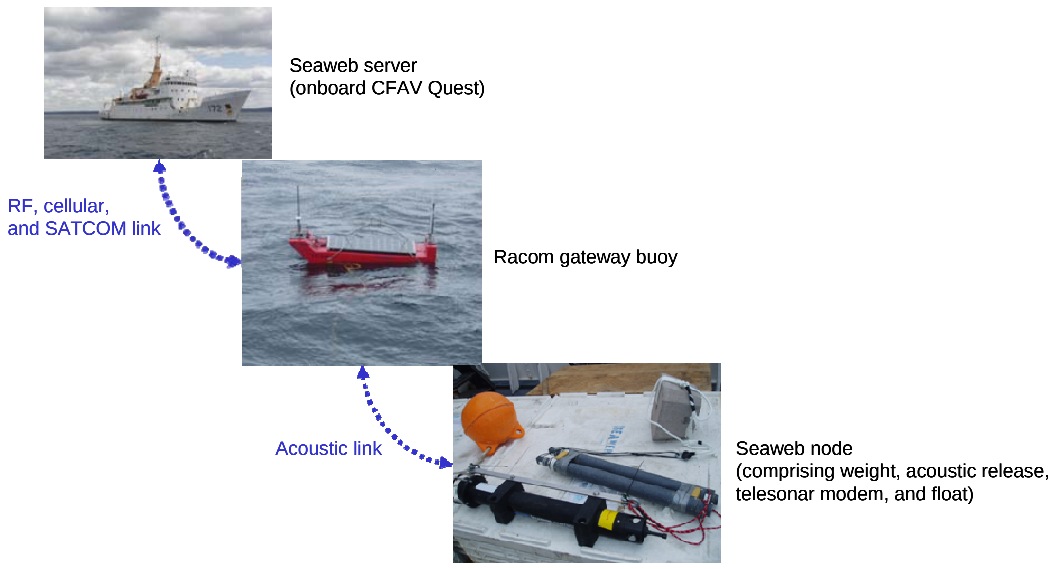


Figure 31. Seaweb network components: Seaweb server, racom gateway buoy, and Seaweb repeater node (after [1]).

Node ID	**Position** (dd^{o} mm.mmm)	**Node ID**	**Position** (dd^{o} mm.mmm)
3	44^{o} 35.609N 63^{o} 59.712W	43	44^{o} 36.713N 63^{o} 58.396W
16	44^{o} 35.400N 63^{o} 59.500W	44	44^{o} 37.072N 63^{o} 58.393W
19	44^{o} 35.279N 64^{o} 00.633W	45	44^{o} 37.347N 63^{o} 58.483W
20	44^{o} 35.870N 63^{o} 59.810W	46	44^{o} 35.639N 63^{o} 59.253W
21	44^{o} 36.350N 63^{o} 59.900W	48	44^{o} 34.302N 63^{o} 59.727W
22	44^{o} 36.850N 63^{o} 59.860W	50	44^{o} 35.747N 64^{o} 00.316W
23	44^{o} 37.340N 63^{o} 59.710W	51	44^{o} 36.468N 64^{o} 00.629W
24	44^{o} 37.810N 63^{o} 59.440W	52	44^{o} 37.097N 64^{o} 00.689W
41	44^{o} 35.790N 63^{o} 58.580W	53	44^{o} 37.694N 64^{o} 00.904W
42	44^{o} 36.270N 63^{o} 58.170W		

Table 8. GPS coordinates of 19 nodes involved in June 2008 Seaweb ad hoc network discovery experiment.

B. ACOUSTIC ENVIRONMENT

Seaweb communications between the acoustic modems occur in the 9-14 kHz frequency band. Environmental observations during the sea trial, permit analysis of the communication channel during the experiment.

1. Transmission Loss

As sound energy propagates away from the source, its intensity decreases due to geometric spreading and attenuation. Geometric spreading is frequency independent and is best described as wavefront expansion as the sound travels away from the source. Initially the expansion begins as spherical spreading with intensity decreasing as a function of the range from the source squared. As the wavefront begins to interact with the seafloor and surface, and the water medium begins to act as a duct, the spreading shifts to a cylindrical model where the intensity decreases in proportion to the range from the source [14].

In addition to geometric spreading, sound energy is also absorbed as it travels through a medium. The attenuation is due to the conversion of the sound energy into heat, and the rate at which this conversion takes place is frequency-dependent. As frequency increases, the rate at which the sound is attenuated increases. Based on work by Francois and Garrison in [17 and 18], the expected rate at which sound attenuation occurs in the ocean at Seaweb frequencies is approximately 1 dB/km. Figure 32 shows the operating frequencies of Seaweb and the corresponding attenuation coefficient.

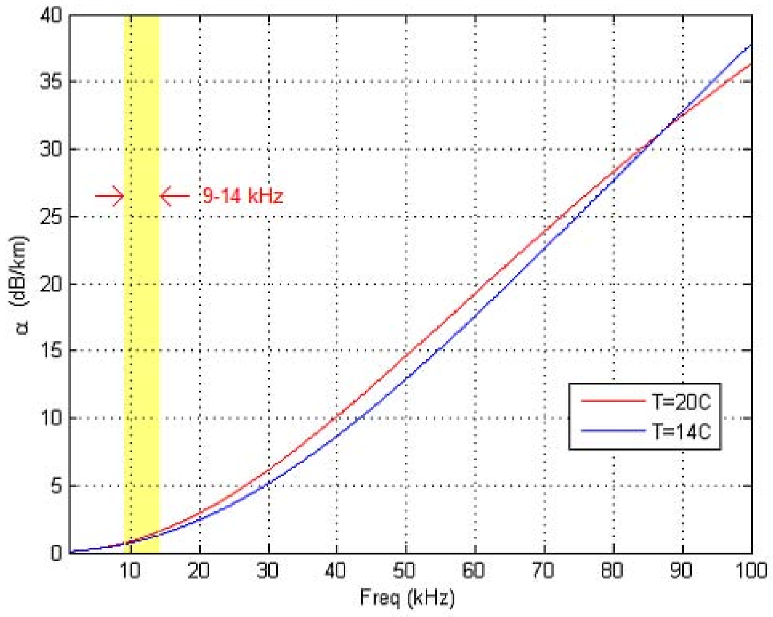


Figure 32. Attenuation coefficient α in dB/km versus transmission frequency in kHz, based on Francois and Garrison [17 and 18] for salinity S = 35 ppt, acidity pH = 8, and depth D = 50 m.

2. Noise Level

The background noise levels in the ocean have several major constituents that each contribute differently depending on the frequency of interest. At very low frequencies, below 20 Hz, the noise is dominated by tidal and wave noise. From 20 Hz to 500 Hz, man-made shipping noise is the main contributor. At 500 Hz, wind noise is dominant until approximately 100 kHz, when thermal noise dictates the background noise levels [14]. Thus, at the Seaweb operating frequencies, wind-driven noise is the main component of the background noise. This is seen in Figure 33 derived from Coates [19].

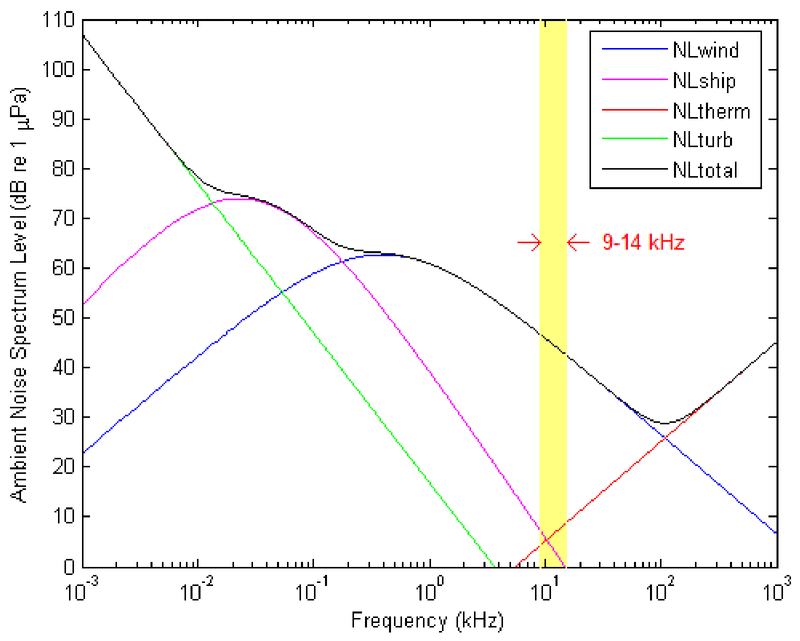


Figure 33. Noise spectrum level based on empirical formula by Coates (after [19]). NLwind is for 5 m/s (10 kts) wind speeds.

Since wind-driven noise is the limiting component of the total background noise in the frequency range of interest for Seaweb, and the wind noise is dependent on the wind speed, Coates' formulas can be expounded upon to give Figure 34. This figure demonstrates the variation in noise level with wind speed. From the figure it is estimated that 50 dB re 1 μPa ambient noise existed at the operating frequencies of Seaweb for the wind speeds observed during the trial.

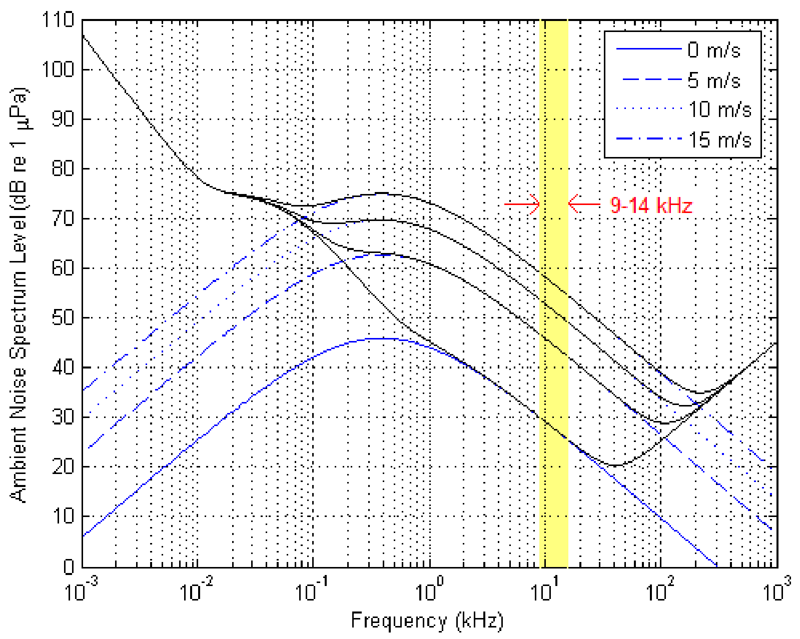


Figure 34. Effect of surface wind speed on noise spectrum level based on empirical formulae by Coates (after [19]).

3. Multipath Propagation

As discussed in Chapter III, the shape of the vertical sound speed profile dictates the path that sound travels from the source. Figure 35 shows the sound-speed profiles taken at the outset and prior to the conclusion of the trial near the gateway node. The average sound-speed profile is analyzed using code from Torres [20] that determines the channel impulse response based on eigenray traces employing a Bellhop Gaussian beam-tracing propagation model. Figure 36 shows the results for nodes at 55 meters in depth at a frequency of 12 kHz, in 57 meters of total water depth. These figures demonstrate that the direct-path arrivals are greater in amplitude than any multipath, and therefore the matched filter used by Seaweb to measure node-to-node ranges should be effective at determining the direct-path arrivals.

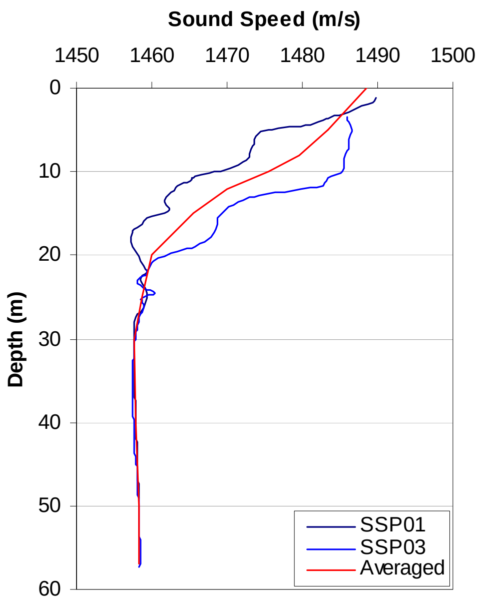


Figure 35. Sound-speed profiles from St. Margaret's Bay.

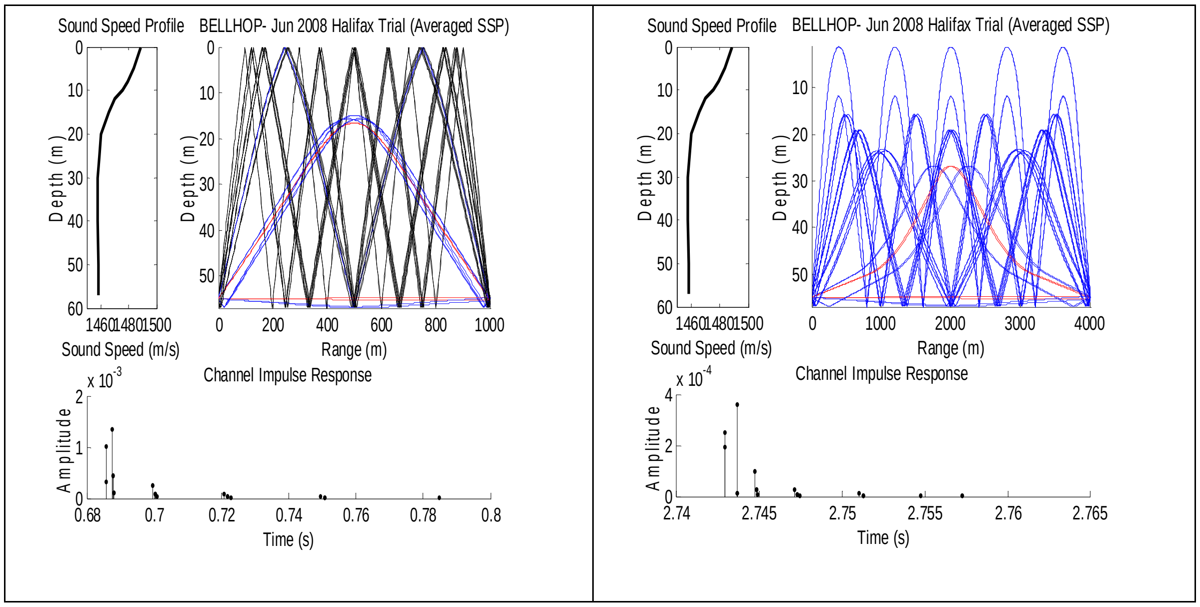


Figure 36. Bellhop eigenray traces for June 2008 St. Margaret's Bay trial show a downward-refracting channel with multipath propagation. Direct-path arrivals are in red. Left: 1 km spacing. Right: 4 km spacing. Note the time-scale and amplitude-scale changes on the channel impulse response plots.

C. DATA

The Seaweb experiments of interest in St. Margaret's Bay were conducted June 23 thru June 25, 2008. During this time the ad hoc discovery method of Ong [1] was exercised on five separate occasions. The node-to-node range measurements were saved into separate files for each discovery process, and all of these files were input into the localization algorithm for analysis. Node 3 was the racom buoy and master node. For the analysis, node 20 was selected as the 'on_X' node for its proximity to the master node and due to it having a large number of neighboring nodes. Figure 37 plots the node locations as determined by GPS at the time of deployment relative to the 'master' node. This figure corresponds to Figure 4, which shows the charted positions of the nodes. Figure 38 plots the same node locations with the axes rotated such that 'on_X' node lies on the *x*-axis.

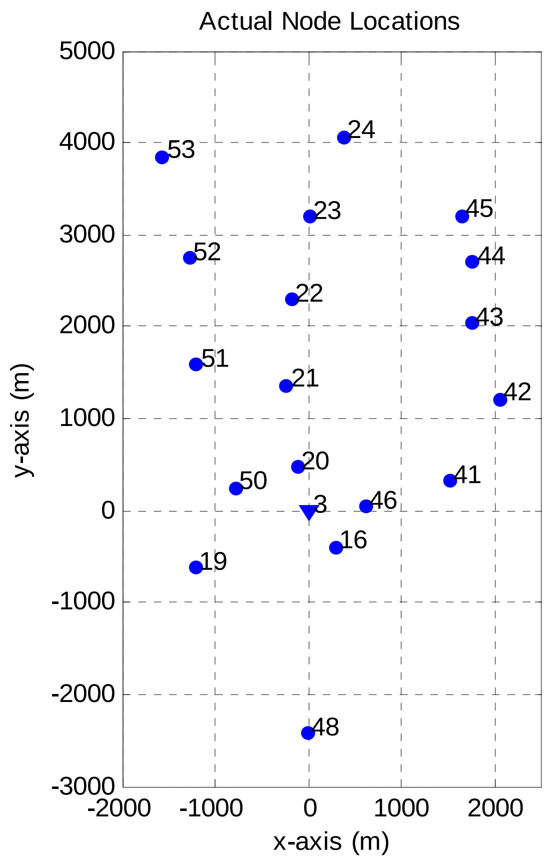


Figure 37. Locations of St. Margaret's Bay nodes referenced to node 3.

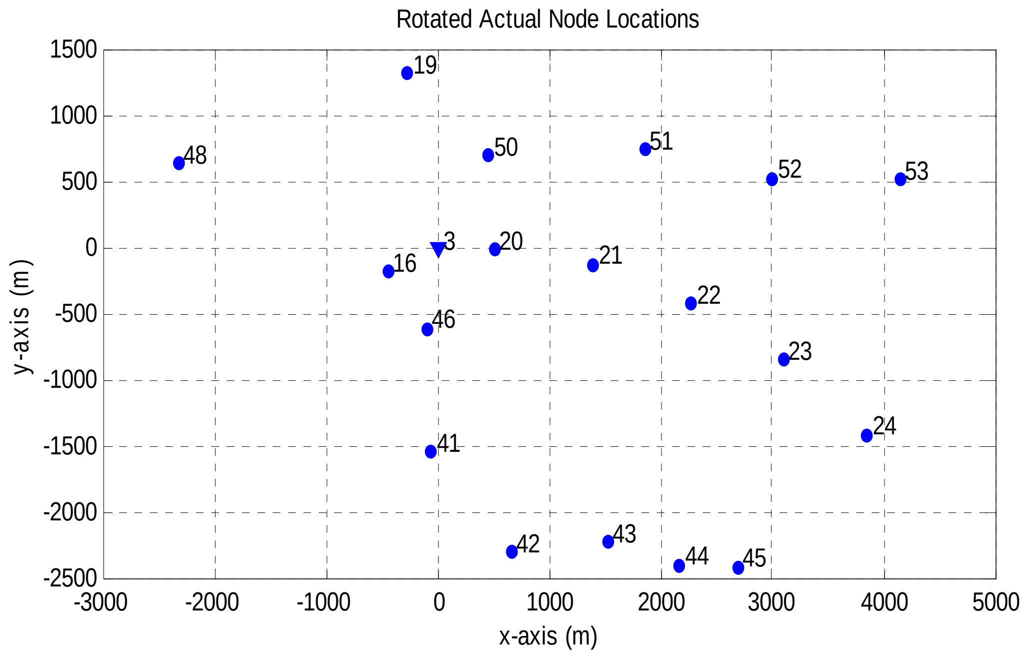


Figure 38. St. Margaret's Bay network rotated with node 3 as the 'master' node and node 20 as the 'on_X' node.

D. RESULTS

Using the range data from the five ad hoc discoveries, the algorithm converges the node locations to within 25 meters in six iterations. The algorithm produces 16 ambiguous solutions. The ambiguous solution that is closest to the actual node positions is solution 71936, which has a mean nodal position error of 80.8 meters. Figure 39 plots the locations of this solution. Appendix B contains plots for all of the ambiguous solutions that are generated.

Additionally an analysis for redundancy is conducted. Using a 25-meter threshold between each node location to call an ambiguous solution redundant, two of the solutions are declared redundant. Namely, ambiguous solution 4096 is redundant to solution 2048 and solution 67480 is redundant to solution 71936, which is the solution with the smallest mean nodal localization error.

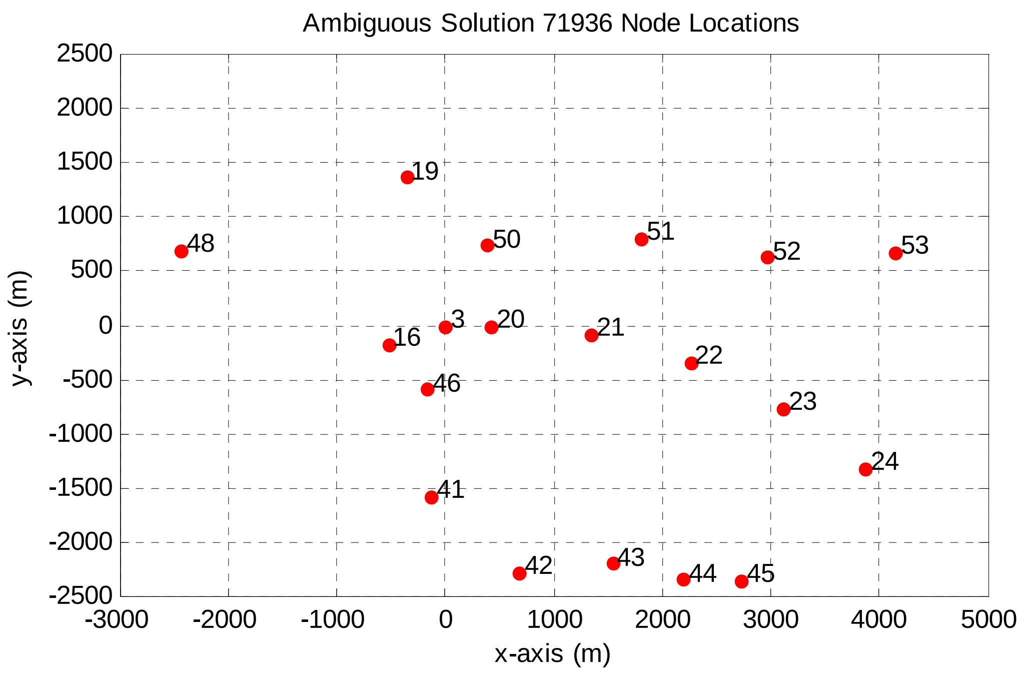
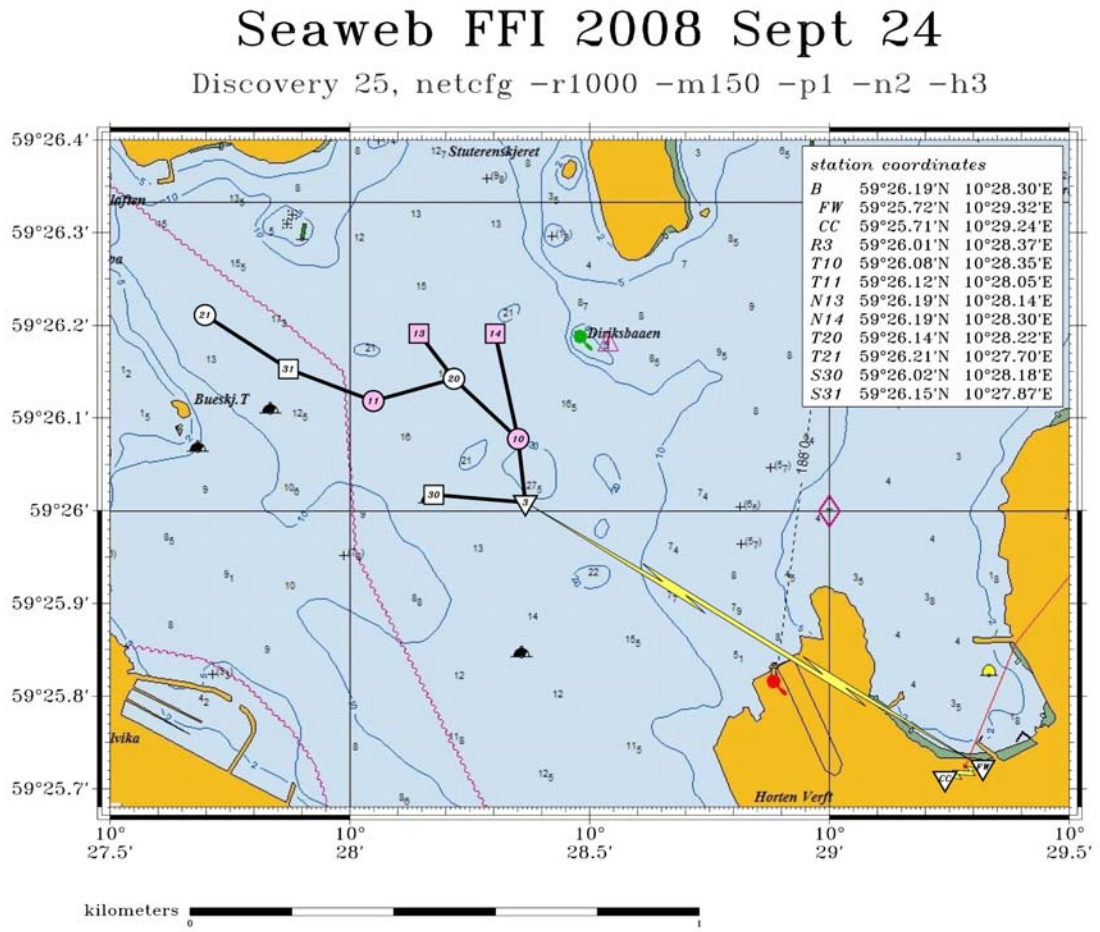


Figure 39. Ambiguous solution exhibiting the smallest mean error per node (80.8 m) compared with the recorded node locations.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. HORTEN EXPERIMENT

A second experiment was conducted off the coast of Horten, Norway in September 2008, but on a smaller scale with fewer nodes and a smaller area. The network deployment area, as seen in Figure 40, was approximately 500 meters by 600 meters and includes nine nodes. Due to the shallow bathymetry and sound speed profile that resulted in most of the sound energy being refracted to the boundaries during the trial, the maximum acoustic range was limited to less than 1500 meters. During the experiment, 32 ad hoc discoveries were conducted over the course of six days. Node 3 was the racom buoy and 'master' node, and node 10 is selected as the 'on_X' node for analysis.



station coordinates

B	59°26.19'N	10°28.30'E
FW	59°25.72'N	10°29.32'E
CC	59°25.71'N	10°29.24'E
R3	59°26.01'N	10°28.37'E
T10	59°26.08'N	10°28.35'E
T11	59°26.12'N	10°28.05'E
N13	59°26.19'N	10°28.14'E
N14	59°26.19'N	10°28.30'E
T20	59°26.14'N	10°28.22'E
T21	59°26.21'N	10°27.70'E
S30	59°26.02'N	10°28.18'E
S31	59°26.15'N	10°27.87'E

Figure 40. Horten Seaweb network layout (from [1]).

The algorithm yields two ambiguous solutions, the better of which has a mean node localization error of 35.7 meters and requires three iterations for the solutions to converge. Figure 41 displays the node locations translated such that node 3 is the origin and node 10 is on the *x*-axis. Figure 42 shows the two ambiguous solutions, of which ambiguous solution 0 is the more accurate one. A visual comparison of solution 0 to the rotated actual node locations reveals that the solution for node 21 is farther away from the 'master' node than the actual location. This follows the general assumption from Chapter III, that the measured ranges will typically exceed the actual ranges, and that these errors compound as the distance from the master node increases.

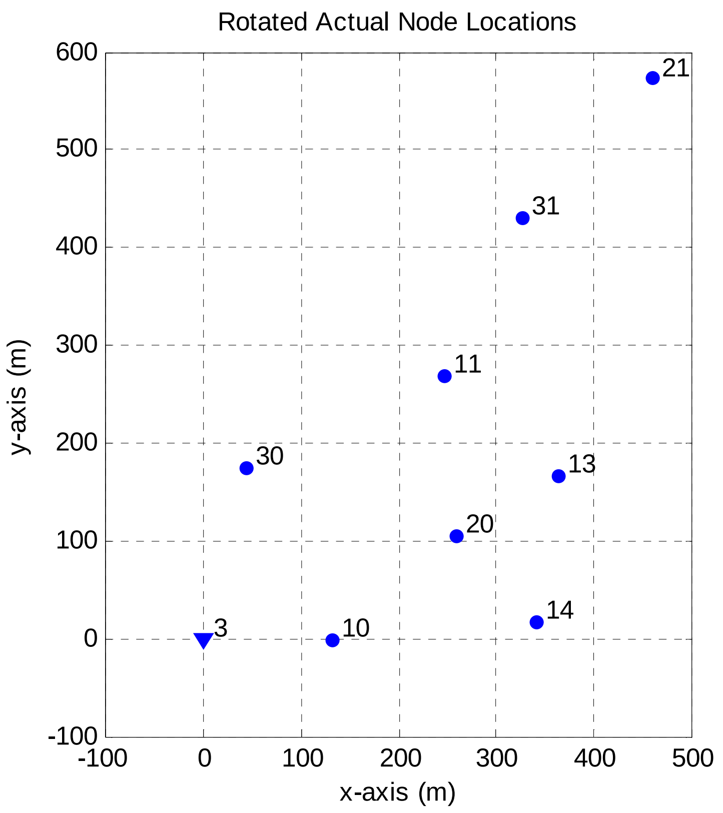


Figure 41. Horten experiment nodes rotated with node 3 as the 'master' node and node 10 as the 'on_X' node.

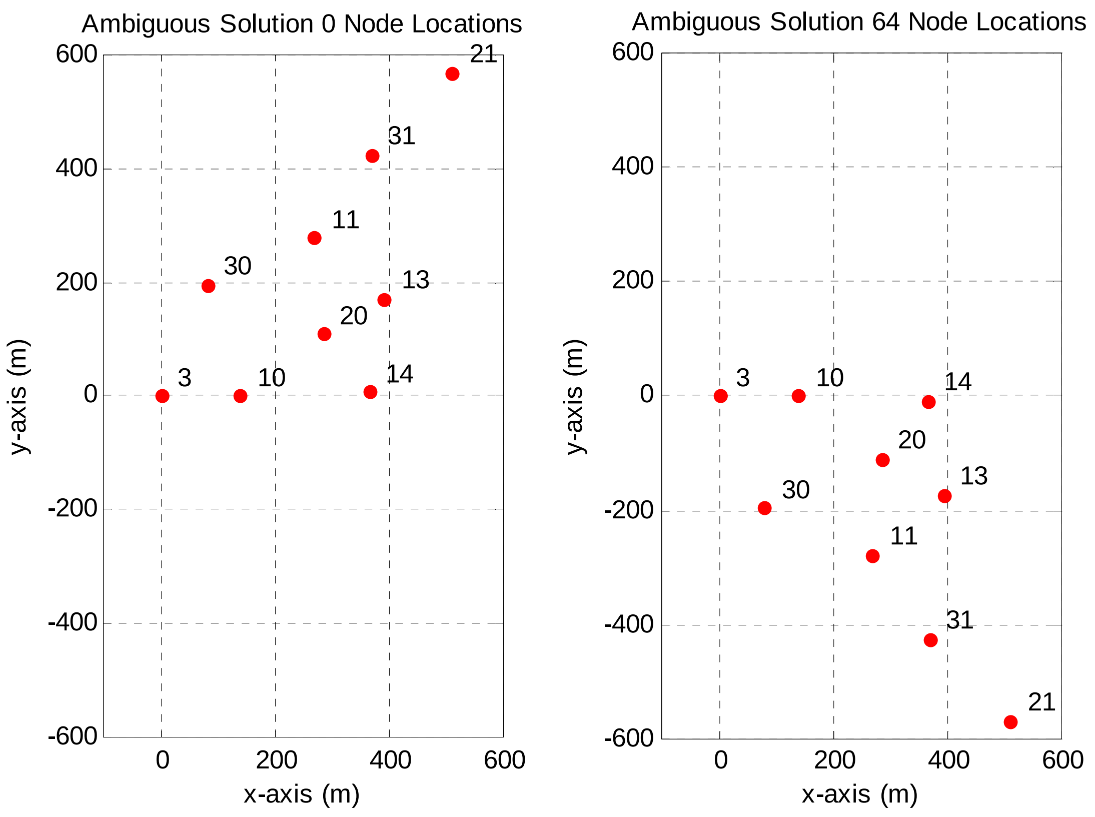


Figure 42. Ambiguous solutions of the Norway sea trial data.

THIS PAGE INTENTIONALLY LEFT BLANK

IX. CONCLUSIONS

A. SUMMARY OF RESULTS

The Seaweb ad hoc discovery process provides node-to-node range measurements that can be used to localize the positions of the network nodes. The results from the synthetic data and experimental data demonstrate that the algorithm implemented in this thesis is capable of localizing the nodes of a Seaweb network. This is especially important for sensor nodes, for which geo-localization is vital to operationally exploit the sensor data gathered.

B. RECOMMENDATION

In view of the fact that the algorithm is capable of localizing nodes, it should be added to the Seaweb field kit. It should be further refined and tested, with the eventual goal of incorporating it into the master node for automated operations.

C. FURTHER RESEARCH

The opportunities for further research stemming from this thesis fall into two broad categories, improving the algorithm to make it more accurate and efficient and having the algorithm interact with the Seaweb network to provide better data input by using system feedback.

1. Pruning Ambiguous Solutions

An improvement would be for the algorithm to selectively determine which unknown node should be evaluated first after the 'master' and 'on_X' nodes have been localized. The candidates for the third node would require ranges to both the 'master' and 'on_X' nodes. The recommended criteria for selecting the best third node from those that qualify would be the node that has the most ranges to other nodes also having ranges to the ‘master’ and ‘on_X’ nodes. The benefit would be that secondary nodes chosen in this manner would not have ambiguous solutions. This would reduce the total number of ambiguous solutions.

Another improvement that would reduce the number of ambiguous solutions is to incorporate operator-provided constraints on the network geometry, such as boundaries, shorelines and known node locations. These constraints could be used to test hypothetical solutions.

2. System Feedback

When the algorithm processes the range data, it is unable to localize any node that has only a single range to known node. To proactively localize these single-range nodes, the algorithm could request of the Seaweb operator that these nodes issue another broadcast ping in an attempt to measure a range to a second known node. The new broadcast ping could be at a higher power level to attempt to overcome background noise, or the system could wait some time interval to allow noise levels and propagation conditions to improve. This same principle could also be applied to nodes having ambiguous solutions. If a third range is found to a known node, the solution becomes exact and eliminates the ambiguity.

The use of a mobile node, such as a UUV or simply a repeater node hung over the side of a ship, provides many avenues for augmenting the range data which are delivered to the algorithm. Assuming the mobile node is capable of determining its own location via GPS, it is an ideal choice as the ‘on_X’ node, because all of the nodes could be geo-localized once the algorithm was complete. The mobile node could also be used to eliminate ambiguous solutions by positioning itself as a known node such that a third range would be available for otherwise ambiguous node solutions. For nodes that have only a single range to a known node, the mobile node could provide ranges from multiple stations. The mobile node could also search for nodes that were deployed but not discovered. To accomplish this, the mobile node could transit to areas where the localization failed to show a node and conduct broadcast pings.

The program Stack_Sift.m could also be modified to detect the migration of nodes over time. Nodes could potentially be moved by bottom-trawling fishermen which would alter the node layout. If the Stack_Sift.m function noted a persistent change in the ranges between nodes, it could either request a complete repeat of the ad hoc discovery process to reset all of the ranges for every node, or it could begin to use the new ranges and disregard the old ones.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A: ALGORITHM

Gen_Solver.m

```
function [b,c,d,e] = Gen_Solver (source)
%LT Dave Zinkhon    22 Sep 08
    %Main Program
    % 02 Oct 08 - Revised to match Soultion_Type_2 using node and range
                % data to determine solution  type to use, and
eliminated
                % the know_third portion
    % 08 Oct 08 - Revised to allow for storage of any possible mirror
solutions
                % that could be encountered

                % Split local_positions into 2 matrixes that could hold
                % 2^(num_nodes-2) possibilities (maximum possible
number of
                % mirrors)

                % Moved master and on_X nodes to top of all matrixes by
                % calling function Reorder

    % 15 Oct 08 - started using active_branches to identify which
splits were
                % being used (identified by decimal number)
    % 18 Feb 08 - revamped to analyze Generated nodes

%clear all
%clc
%close all

global X_positions;
global Y_positions;
global ranges;
global num_of_nodes;
global master_branch;
global active_branches;

ranges = source;

%seeds active_branches with first branch (0)
active_branches = 0;


%determine number of nodes for loop control
num_of_nodes = length (ranges(1,:))-1;

%create array for storage of positions in local reference frame
    %1st column is node number, each column is then possibility for
mirror
```

```
X_positions = zeros (num_of_nodes+1 , 2^(num_of_nodes-2)+1);

for i = 1:num_of_nodes
    X_positions (i+1,1) = ranges (i+1,1);
end

%fills in unknown data and column headers with NaN's
for i = 2:length(X_positions(1,:))
    X_positions (:,i) = NaN;
end
X_positions(1,1) = NaN;

%Y_position matrix is identical in setup to X_positions
Y_positions = X_positions;

%this is used to track which branches are currently being used and
which
    %branches even exist
master_branch = zeros (num_of_nodes+1, 2);
master_branch(:,1) = X_positions(:,1);
master_branch(1,2) = NaN;

%set master node to origin and on_X position to range from master to
on_X
    %master
X_positions (2,2) = 0;
Y_positions (2,2) = 0;
    %on_X
X_positions (3,2) = ranges (3,2);
Y_positions (3,2) = 0;



%change_local used to keep program running if the has been a new node
    %solved in the local positions matrix
change_local = 1;

%passes used to track the number of times the program loops to
determine
    %local_positions with precision of .01 meters (note: this may not
be
    %accurate to actual positions)
%%
passes = 0;

while change_local>0 && passes < 100
        change_local = 0;
        passes = passes+1;

    for n = 4:num_of_nodes+1

%determine which solution type to use (Matrix or Law of Cosines)

    soln = Solution_Type_3(n);
```

```
            if soln <= 1       %insufficient range data for solution
               % disp (['Pass: ',num2str(passes),', For node: ',
num2str(X_positions (n,1))])
               % disp('Cannot solve this node due to an insufficient
number of ranges.')


            elseif soln == 2     %mirror image due to only two ranges
               % disp (['Pass: ',num2str(passes),', For node: ',
num2str(X_positions (n,1))])
               % disp ('This point has a mirror image due to only
having two ranges.')
                change_local = change_local + Cosines_Approach_3 (n);

               % X_positions
               % Y_positions
               % active_branches

            else    %enough data to find location exactly
                change_local = change_local + Matrix_Approach_3 (n);

               % X_positions
               % Y_positions
               % active_branches

            end

    end
   disp(['Completed pass: ',num2str(passes)])
end

b = X_positions;
c = Y_positions;
d = active_branches;
e = passes;
```

Stack_Build.m

```
function b = Stack_Build (nodes, master, on_x, source)
% LT Dave Zinkhon   13 Jan 2009
    %program to load data layers a place in stack for reduction by
Stack_Sift

%Inputs
%   nodes - vector numbered addresses of all nodes in the field
%   master - address of node that will be used as geographic reference
for
%            solutions.  All solutions will have this node located at
(0,0)
%   on_x - address of node that will placed at a known bearing from
master.
```

```
%           All solutions will have this node at (Range (master-
>on_x),0)
%    source - matrix containing names of .txt files containing range
data

%Output
%    b - 3-D stack of node-to-node range data that can then be analyzed
to
%        eliminate outliers in the data (Stack_Sift)

num_nodes = length(nodes);  %number of nodes deployed (allows creating
array beforehand)
num_files = length(source(:,1));   %number of files to be loaded
                                   %Same as numbers of layers in the
cube
template = zeros (num_nodes+1);     %template will be used to ensure
each layer of stack matches
Stack = zeros (num_nodes+1,num_nodes+1,num_files);

for i = 1:num_nodes     %places node names in row and column headers
    template (1,i+1) = nodes(i);
    template (i+1,1) = nodes(i);
end

%move master to front position of rows & columns
    loc = find (template(1,:)== master); %returns column index of
master node
    template (1,loc) = template (1,2);  %moves node in first position
to master's position
    template (loc,1) = template (2,1);  %same for rows
    template (1,2) = master;    %moves master to 1st position
    template (2,1) = master;    %same for rows

%move on_x to 2nd position in rows and columns
    loc = find (template(1,:)== on_x); %returns column index of on_x
node
    template (1,loc) = template (1,3);  %moves node in first position
to on_x's position
    template (loc,1) = template (3,1);  %same for rows
    template (1,3) = on_x;    %moves on_x to 1st position
    template (3,1) = on_x;    %same for rows

for d = 1:num_files
    name = source(d,:);
    data = load (name);       %name of file to be loaded (must be full)

    layer = template;

    for i = 1:length(data(:,1))
     src = data(i,1);     %determines source node
        for j = 2:length(data(1,:))
            if rem(j,2)==0
                rcvr = data(i,j);    %determines receiver node
                rng = data (i,j+1);    %determines range between source
and rcvr

 %find proper row and column in ranges to store data
                row = find(layer(:,1)== src);
                col = find(layer(1,:)== rcvr);
                if rcvr == 0    %prevents overwriting row headers
                    row = 1;
                    col = 1;
                end
                layer(row,col) = rng;   %store range data
            end
        end
    end

    Stack(:,:,d) = layer;
    %Stack(1,1,d) = d;
        %range cube has raw ranges from each data run stacked over time
end

b = Stack;
```

Stack_Sift.m

```
function b = Stack_Sift (Stack)
% LT Dave Zinkhon   13 Jan 2009
    %program to load remove extraneous data from node to node ranges in
        %Stack passed in
    %weight array is used to zero weight any data that is outside 10%CI
of mean (1.645 standard deviations)
    %For the output, a NaN will appear in any Range slot that does not
have any good ranges including along the diagonal


num_nodes = length(Stack(1,:,1))-1; %number of nodes in field
num_files = length(Stack(1,1,:));   %height of data stack

ranges = zeros(2* num_files,1);     %stores node to node ranges down
stack
weights = zeros(2* num_files,1);    %stores weights for data removal (0
or 1)

Good_Ranges = zeros(num_nodes+1, num_nodes+1);  %output matrix
Good_Ranges(1,:) = Stack (1,:,1);
Good_Ranges(:,1) = Stack (:,1,1);
%count = 0

%Portion for sorting bad ranges out of good for entire cube
    %Good_Ranges will contain mean of usable ranges for output to
Main.m
for i = 2:num_nodes+1
    for j = 2:num_nodes+1
```

```
        for d = 1:num_files     %retrieve all data
            ranges(d) = Stack (i,j,d);  %stores a->b range
            ranges(length(ranges)-d+1) = Stack(j,i,d);  %stores b->a
range
        end

        for d = 1:length(weights)
            if ranges(d) > 1    %assign weight if positive range exists
                weights(d) = 1;
            else
                weights(d) = 0;
            end
        end

        worst = .00001;   %net > zero to get initial entry to loop
        worst_loc = 0;
        while worst == .00001
            rXw = ranges .* weights;
            mean = sum(rXw)/sum(weights);   %mean of ranges still in
use

            tot = 0;
            for a = 1:length(ranges)
                tot = tot + ((ranges(a) - mean)^2) * weights(a);
            end

            dev = sqrt(tot/(sum(weights)-1));   %stand dev of ranges
still in use

            for a = 1:length(ranges)
                if weights(a) == 1  %only look at ranges still being
weighted
                    if abs(ranges(a)-mean) > 1.645*dev  %check if
outside CI
                        if abs(ranges(a)-mean) > worst   %check to see
if furthest outside mean
                            worst = abs(ranges(a)-mean);
                            worst_loc = a;
                        end
                    end
                end
            end

            if worst_loc > 0    %eliminate range outside CI
                weights(worst_loc) = 0; %set weight to zero (removes
range)
                worst_loc = 0;  %resets worst location
                worst = .00001; %ensures will run through while loop
again
            else
                worst = 0;  %no range outside CI => exist while loop
            end
        end
        Good_Ranges(i,j) = mean;    %stores mean of remaining ranges
        Good_Ranges(j,i) = mean;
        Good_Ranges(1,1) = NaN;     %puts a Nan in upper left
  %count = 0
    end
end

b = Good_Ranges;
```

Data Analysis / Creation

Halifax_Analysis

```
%LT Dave Zinkhon    8 Mar 2008
    %Halifax Analysis
    %Used to find solutions to Halifax ad hoc data
    %The quantities saved for each iteration are:
        %number of passes required to get results
        %number of nodes that have solutions
        %number of ambiguous solutions
        %smallest total range difference between actual node locations
and
            %the solution for any ambiguous solution


close all
clc

r=1;

Stack = Stack_Build(in_nodes, master, on_X, Halifax_Source);

Shift_Result = Stack_Sift (Stack);

[X_positions, Y_positions, active_branches, passes] = Gen_Solver
(Shift_Result);


%Statistic Collection for number of passes, number of ambiguous
        %solutions, number of nodes not localized
tot_passes(r) = passes;

NaN_nod_sol = 1-isnan(X_positions(:,2));
nod_sol(r) = sum(NaN_nod_sol);

ambig(r) = length(active_branches);
```

```
%%%%%Comparison of solutions to Actual Locations
Rotate_Original_Nodes   %rotates source nodes for comparison (master =
0,0...

%take only non-NaN solutions from X_positions and Y_positions
X_quick_ref = zeros(length(X_positions(:,1)),
length(active_branches)+1);
X_quick_ref(:,1) = X_positions(:,1);

Y_quick_ref = X_quick_ref;

for i = 1:length(active_branches)
   col =  active_branches(i)+2;
   X_quick_ref(:,i+1) = X_positions(:,col);
   Y_quick_ref(:,i+1) = Y_positions(:,col);

end



for i = 2:length(active_branches)+1
    er_tot = 0;
    NaN_branch = 1-isnan(X_quick_ref(:,i));
    for j = 2:length(X_quick_ref(:,1))

        if NaN_branch(j) == 1  %find tot range error from actual to
active branch
            x_er = abs(rotated_nodes(j-1,2)-X_quick_ref(j,i));
            y_er = abs(rotated_nodes(j-1,3)-Y_quick_ref(j,i));
            er_tot = er_tot + sqrt (x_er^2 + y_er^2);
        end



    end
    min_er(i-1) = er_tot;
end

    Best_Error(r) = min(min_er);

%%%%%Attempt to remove redundant ambiguous solutions
redundant = 0;
for i = 2:length(X_quick_ref(1,:))
    not_redundant = 0;
    for k = i+1:length(X_quick_ref(1,:))
        for j = 2:length(X_quick_ref(:,1))
            if (abs(X_quick_ref(j,i) - X_quick_ref(j,k)) >=17) &&
(abs(Y_quick_ref(j,i) - Y_quick_ref(j,k)) >=17)
                not_redundant = not_redundant + 1;
            end
        end

        if not_redundant < 1 && i~=k
            redundant(i,k) = 1;
        else
```

```
            redundant(i,k) = 0;
        end
    end
end

tot_redundant(r) = sum(sum(redundant));
```

Error_Free

```
%LT Dave Zinkhon    8 Mar 2008
    %Error Free
    %Used to rune multiple iterations of error free generated nodes for
        %statistical analysis
    %The quantities saved for each iteration are:
        %number of passes required to get results
        %number of nodes that have solutions
        %number of ambiguous solutions
        %smallest total range difference between actual node locations
and
            %the solution for any ambiguous solution

clear all
close all
clc

runs = 50;     %number of node sets that are to be analyzed

for r = 1:runs
    r
    %program that will create nodes for analysis
    Point_Gen_Gaus_for_Stack

    Shift_Result = Stack_Sift (Stack);

    [X_positions, Y_positions, active_branches, passes] = Gen_Solver
(Shift_Result);


    %Statistic Collection for number of passes, number of ambiguous
        %solutions, number of nodes not localized
    tot_passes(r) = passes;

    NaN_nod_sol = 1-isnan(X_positions(:,2));
    nod_sol(r) = sum(NaN_nod_sol);

    ambig(r) = length(active_branches);


%%%%%Comparison of solutions to Actual Locations
    Rotate_Original_Nodes   %rotates source nodes for comparison
(master = 0,0...
    %take only non-NaN solutions from X_positions and Y_positions
    X_quick_ref = zeros(length(X_positions(:,1)),
length(active_branches)+1);
    X_quick_ref(:,1) = X_positions(:,1);

    Y_quick_ref = X_quick_ref;

    for i = 1:length(active_branches)
        col =  active_branches(i)+2;
        X_quick_ref(:,i+1) = X_positions(:,col);
        Y_quick_ref(:,i+1) = Y_positions(:,col);

    end


    for i = 2:length(active_branches)+1
        er_tot = 0;
        NaN_branch = 1-isnan(X_quick_ref(:,i));
        for j = 2:length(X_quick_ref(:,1))

            if NaN_branch(j) == 1  %find tot range error from actual to
active branch
                x_er = abs(rotated_nodes(j-1,2)-X_quick_ref(j,i));
                y_er = abs(rotated_nodes(j-1,3)-Y_quick_ref(j,i));
                er_tot = er_tot + sqrt (x_er^2 + y_er^2);
            end

            %if er_tot < min_er  %store smallest total error for any
ambiguous soln
            %    min_er = er_tot;
            %    soln = i;
            %end
        end
        min_er(i-1) = er_tot;
    end

    Best_Error(r) = min(min_er);


end
```

Point_Gen_for_Stack

```
%LT Dave Zinkhon    7 Aug 08
    %Creating points in Range Area
    %Determining ranges between points
    %Plotting points within area

    %22 Aug 08 - changed ranges format to match format created when
inputting data
```

```
    %24 Sep 08 - updated to put Gaussian (mu,s) range error in ranges
    %17 Feb 09 - changed to create stack of ranges for testing sifter
    %09 Mar 09 - changed how randomness was added to make
        %   r(report) = r(actual)*a + b
        %   a = normrnd(1,.1)
        %   b = b = (.05*ranges)*chi2rnd(1) note:chisqr(1) has mean of
1

w = 5000;   %width of area (x) in meters
l = 5000;   %length of area (y) in meters
num_of_nodes = 5;     %number of nodes in area
mu = 1;    %mean of range offset
s = .1;    %standard dev of offset
layers = 50;    %number of routes run by ad-hoc network
max_range = 4000;   %maximum range that would be calculated/received by
network

nodes = zeros(num_of_nodes,3);      %array for storing node location

for i = 1:num_of_nodes        %loop to put node #, x-location, y-
location
    nodes(i,1) = i;
    nodes(i,2) = unidrnd(w);
    nodes(i,3) = unidrnd(l);
end

ranges = zeros (num_of_nodes+1);    %array for storing node to node
ranges

for i = 1:num_of_nodes     %makes row and column headers
    ranges (i+1,1) = i;
    ranges (1,i+1) = i;
end

%loop to determine ranges, normrnd component will result in range from
1 to 2
    %to be different from 2 to 1 to simulate differences expected in
real data
for i = 1:num_of_nodes
    for j = 1:num_of_nodes
        ranges(i+1,j+1) = sqrt((nodes(i,2)-nodes(j,2))^2+(nodes(i,3)-
nodes(j,3))^2);
    end
end

for i = 1:length(ranges(1,:))   %places NaN's along diagonal
    for j = 1:length(ranges(:,1))
        if i==j
            ranges (i,j) = NaN;
        end
    end
end

%figure (1)      %plots each individual point within range specified
```

```
%    plot (nodes(:,2),nodes(:,3), 'bo');
%   axis([0,w,0,l])

%Build Stack here

Stack = zeros (num_of_nodes+1,num_of_nodes+1,layers);

%put headers on each column/row
for k = 1:layers
    Stack(1,:,k) = ranges (1,:);
    Stack(:,1,k) = ranges (:,1);
end

for k = 1:layers
    for i = 1:num_of_nodes
        for j = 1:num_of_nodes
            Stack(i+1,j+1,k) =
ranges(i+1,j+1)*normrnd(mu,s)+(.05*ranges(i+1,j+1))*chi2rnd(1);

            if Stack(i+1,j+1,k) > max_range
                Stack(i+1,j+1,k) = NaN;
            end

            %ensures that range exists between master and on_X
            if i+1==3 && j+1==2
                Stack (i+1,j+1,k) =
ranges(i+1,j+1)*normrnd(mu,s)+(.05*ranges(i+1,j+1))*chi2rnd(1);
            end

            if i+1==2 && j+1==3
                Stack (i+1,j+1,k) =
ranges(i+1,j+1)*normrnd(mu,s)+(.1*ranges(i+1,j+1))*chi2rnd(1);
            end

        end
    end
end


clear l w i j num_of_nodes a b mu s k layers max_range
```

Rotate_Original_Nodes.m

```
%LT Dave Zinkhon    22 OCT 08

%%
%Translating and rotating generated points to local axis

master = 3; %added for generated points
on_X = 20;
```

```
m = find (nodes(:,1) == master);
p = find (nodes(:,1) == on_X);

x_off = nodes(m,2);
y_off = nodes(m,3);

offset_nodes = nodes;

num_of_nodes = length(nodes(:,1));


ranges = zeros (num_of_nodes+1);    %array for storing node to node
ranges

for i = 1:num_of_nodes     %makes row and column headers
    ranges (i+1,1) = in_nodes(i);
    ranges (1,i+1) = in_nodes(i);
end

for i = 1:num_of_nodes        %loop to determine ranges
    for j = 1:num_of_nodes
        ranges(i+1,j+1) = sqrt((nodes(i,2)-nodes(j,2))^2+(nodes(i,3)-
nodes(j,3))^2);
    end
end

for i = 1:length(ranges(1,:))   %replaces 0's with NaN's
    for j = 1:length(ranges(:,1))
        if ranges (i,j) == 0
            ranges (i,j) = NaN;
        end
    end
end

%%

for i = 1:length(nodes(:,1))
   offset_nodes (i,2) = nodes(i,2) - x_off;
   offset_nodes (i,3) = nodes(i,3) - y_off;
end
%%
opp = offset_nodes(p,3);
adj = offset_nodes(p,2);

%possible scenarios for quadrants for on_X position to master

rot = atan2(opp,adj);
if rot < 0
    rot = rot+2*pi;
end

rotated_nodes = offset_nodes;
```

```
for i = 2:length(nodes(:,1))
   theta_o = atan2(offset_nodes(i,3),offset_nodes(i,2));
   if theta_o < 0
       theta_o = theta_o + 2*pi;
   end
   rotated_nodes (i,2) = ranges(i+1,2)*cos(theta_o-rot);
   rotated_nodes (i,3) = ranges(i+1,2)*sin(theta_o-rot);
end

figure (2)      %Given (known) positions
for i = 1:length(rotated_nodes)
    if i == 1
        plot (rotated_nodes(i,2),rotated_nodes(i,3),
'bv','MarkerFaceColor',[0 0 1]);
        node_name = num2str(rotated_nodes(i,1));
        text(rotated_nodes(i,2)+10,rotated_nodes(i,3)+10,{node_name});
    else
    plot (rotated_nodes(i,2),rotated_nodes(i,3), 'b.',
'MarkerSize',15);
    node_name = num2str(rotated_nodes(i,1));
    text(rotated_nodes(i,2)+10,rotated_nodes(i,3)+10,{node_name});
    end

    hold on
end
title ('Rotated Actual Node Locations');
xlabel ('x-axis (m)');
ylabel ('y-axis (m)');
grid on
axis equal tight
axis([-3000, 5000, -2500, 1500])

hold off

clear p x_off y_off offset_nodes opp adj rot
```

Functions called by other programs

Reorder_3.m

```
function Reorder_3 (master, on_X)

%LT Dave Zinkhon    8 OCT 08
    %This function will move the master and on_X nodes to the top of
all
    %matrixes to ease understanding and reduce loop iterations required
in
    %the Main_3

%20 Oct 08 - Revise due to error found if one of "known" nodes was
already
            %in first two rows
```

```
global ranges;

%stores data in first rows
store_1st = ranges(2,:);

%finds which rows previously held master node
m = find (ranges(:,1) == master);

%move master and on_X ranges data to top row
ranges(2,:) = ranges(m,:);

%move top row data to original master row
ranges(m,:) = store_1st;

%move column order to match row order

%stores data in first and second columns
store_1st = ranges(:,2);

%move master and on_X ranges data to 1st two columns
ranges(:,2) = ranges(:,m);

%move top two row data to original master and on_X rows
ranges(:,m) = store_1st;

%Repeat for on_X node
store_2nd = ranges(3,:);
n = find (ranges(1,:) == on_X);
ranges(3,:) = ranges(n,:);
ranges(n,:) = store_2nd;
ranges(:,3) = ranges(:,n);
store_2nd = ranges(:,3);
ranges(:,n) = store_2nd;
```

Solution_Type_3.m

```
function solution = Solution_Type_3 (n)
%LT Dave Zinkhon    19 SSEP 08
    %writing logic to determine if law of cosines should be used or
Matrix
        %Solution should be used
    %O/P is matrix stating which type of method should be used to find
location
    %2 Oct 08 - Modified to test based on known nodes and ranges vice
just
                %known ranges, For use in Main_2
    %8 Oct 08 - Modified for Main_3 to match new data structure

global ranges;
global X_positions;

%solution = <1 means cannot determine anything (single range available)
%solution = 2 means have two ranges and will use law of cosines (Find
Locally)
%solution >= 3 means have 3 or more ranges and will use matrix solution

R = isnan(ranges(:,n));%logic test for NaN's in ranges
LN = isnan(X_positions(:,2));%logic test for NaN's in nodes
corr = R+LN;    %if result is zero, means is range to known position

usable = find (corr==0);    %array that know which nodes are usable

solution = length (usable);
```

Bin_to_Dec.m

```
function dec = Bin_to_Dec_Converter (branch)

%LT Dave Zinkhon     8 Oct 08

    %This function will take the position within a branch
        %variables and convert them into a base10 number that will be
used
        %to store the node locations in the proper column of
X_positions
        %and Y_positions

global master_branch;

%b is used to reverse value of exponent as i increases
b = length (master_branch(:,1));

dec = 0;
for i = 2:length (master_branch(:,1))
    a = branch(i,2) * (2^(b-i));
    dec = dec + a;
end
```

Dec_to_Bin.m

```
function bin = Dec_to_Bin_Converter (dec)

%LT Dave Zinkhon     8 Oct 08

    %This function will take the position within a branch
        %variables and convert them into a base10 number that will be
used
        %to store the node locations in the proper column of
X_positions
```

```
        %and Y_positions

global master_branch;

%b is used to reverse value of exponent as i increases
b = length (master_branch(:,1));

bin = master_branch;

for i = 2:length (master_branch(:,1))
    if (2^(b-i)) <= dec     %places a one in appropriate node
        dec = dec - 2^(b-i);
        bin (i, 2) = 1;
    else
        bin (i, 2) = 0;     %else place a zero (ensures overwriting)
    end
end
```

Cosines_Approach_3.m

```
function change = Cosines_Approach_3 (n)
%%
%LT Dave Zinkhon    8 Aug 08
    %Takes ranges from Point_Gen plots them in local reference frame
    %19 Sep 08 - Modified to only work for 2 known ranges, Matrix
approach
                %will be taken for 3 known ranges solution (Matrix
                %Approach)
    %22 Sep 08 - Removed plotting function and moved to
Martix_Approach.m)
    %24 Sep 08 - formatted to function for use in Main_1
    %15 Oct 08 - altered for use in Main_3, using branch tracking,
                %separate X and Y position tracking
    %23 Oct 08 - replaced elseif atan portion with atan2 function

global X_positions;
global Y_positions;
global ranges;
global master_branch;
global active_branches;

%length is determined prior to executing loop to prevent unnecessary
    %looping if active_branches is added to during loop
length_active_branches = length(active_branches);

for i = 1:length_active_branches
%%
%points to proper column for given branch number
    branch_col = active_branches(i)+2;      %+2 aligns to proper column
number
    branch = Dec_to_Bin_Converter (active_branches(i));
%%
```

```
%Logic tests to determine which nodes are being referenced
    R = isnan(ranges(:,n));%logic test for NaN's in ranges
    LN = isnan(X_positions(:,branch_col));%logic test for NaN's in
nodes
    corr = R+LN;    %if result is zero, means is range to known
position

    usable = find (corr==0);    %array that know which nodes are usable

%%
if length(usable)>=2

%used to track when changes are made to position matrixes
    prior_x = X_positions (n,branch_col);
    prior_y = Y_positions (n,branch_col);
%%
%Calculation of what rotation is required about local axis
    if X_positions(usable(2),branch_col) >
X_positions(usable(1),branch_col) && Y_positions(usable(2),branch_col)
> Y_positions(usable(1),branch_col)
        theta_axis = atan((Y_positions(usable(2),branch_col) -
Y_positions(usable(1),branch_col))/(X_positions(usable(2),branch_col) -
X_positions(usable(1),branch_col)));
    elseif X_positions(usable(2),branch_col) <
X_positions(usable(1),branch_col) && Y_positions(usable(2),branch_col)
> Y_positions(usable(1),branch_col)
       theta_axis = pi() + atan((Y_positions(usable(2),branch_col) -
Y_positions(usable(1),branch_col))/(X_positions(usable(2),branch_col) -
X_positions(usable(1),branch_col)));
    elseif X_positions(usable(2),branch_col) <
X_positions(usable(1),branch_col) && Y_positions(usable(2),branch_col)
< Y_positions(usable(1),branch_col)
         theta_axis = pi() + atan((Y_positions(usable(2),branch_col) -
Y_positions(usable(1),branch_col))/(X_positions(usable(2),branch_col) -
X_positions(usable(1),branch_col)));
    elseif X_positions(usable(2),branch_col) >
X_positions(usable(1),branch_col) && Y_positions(usable(2),branch_col)
< Y_positions(usable(1),branch_col)
        theta_axis = atan((Y_positions(usable(2),branch_col) -
Y_positions(usable(1),branch_col))/(X_positions(usable(2),branch_col) -
X_positions(usable(1),branch_col)));
    elseif X_positions(usable(2),branch_col) ==
X_positions(usable(1),branch_col) && Y_positions(usable(2),branch_col)
> Y_positions(usable(1),branch_col)
        theta_axis = pi()/2;
    elseif X_positions(usable(2),branch_col) ==
X_positions(usable(1),branch_col) &&
Y_positions(usable(2),branch_col)<Y_positions(usable(1),branch_col)
        theta_axis = -pi()/2;
    elseif X_positions(usable(2),branch_col) >
X_positions(usable(1),branch_col) && Y_positions(usable(2),branch_col)
== Y_positions(usable(1),branch_col)
        theta_axis = 0;
    else
        theta_axis = -pi();
    end
%%
%This is where calculation is done based on law of cosines to find
third
    %point
top = ranges(usable(1),n)^2 + ranges(usable(1),usable(2))^2 -
ranges(usable(2),n)^2;
bot = 2 * ranges(usable(1),n) * ranges (usable(1),usable(2));
theta_tri = acos (top/bot);

%%
    if branch(n,2) == 0
%Left side solution
        X_positions (n,branch_col) = X_positions (usable(1),branch_col)
+ ranges(usable(1),n) * cos(theta_axis+theta_tri);
        Y_positions (n,branch_col) = Y_positions (usable(1),branch_col)
+ ranges(usable(1),n) * sin(theta_axis+theta_tri);

%now shift to new alternative column (if applicable) for right side
solution
        branch(n,2) = 1;
        new_branch_num = Bin_to_Dec_Converter(branch);
        prior = find(active_branches == new_branch_num);
            if prior  %logic test to see if prior is empty array (if
branch already exists, do nothing)
            else    %adds new branch number to active_branches if
branch not already exist
                active_branches(length(active_branches)+1) =
new_branch_num;
            end
        new_branch_col = new_branch_num+2;
%Update branch solution sets
        master_branch (n,2) = 1;
%copy previous branch values to new column
        X_positions (:,new_branch_col) =  X_positions (:,branch_col);
        Y_positions (:,new_branch_col) =  Y_positions (:,branch_col);
%if 3rd node is to right side =>theta_tri is subtracted
        X_positions (n,new_branch_col) = X_positions
(usable(1),new_branch_col) + ranges(usable(1),n) * cos(theta_axis-
theta_tri);
        Y_positions (n,new_branch_col) = Y_positions
(usable(1),new_branch_col) + ranges(usable(1),n) * sin(theta_axis-
theta_tri);
    end
%%
%test to see if positions were updated
    if abs(prior_x - X_positions (n,branch_col)) <1 && abs(prior_y -
Y_positions (n,branch_col))<1
        change = 0;
    else
        change = 1;
    end
end
end
```

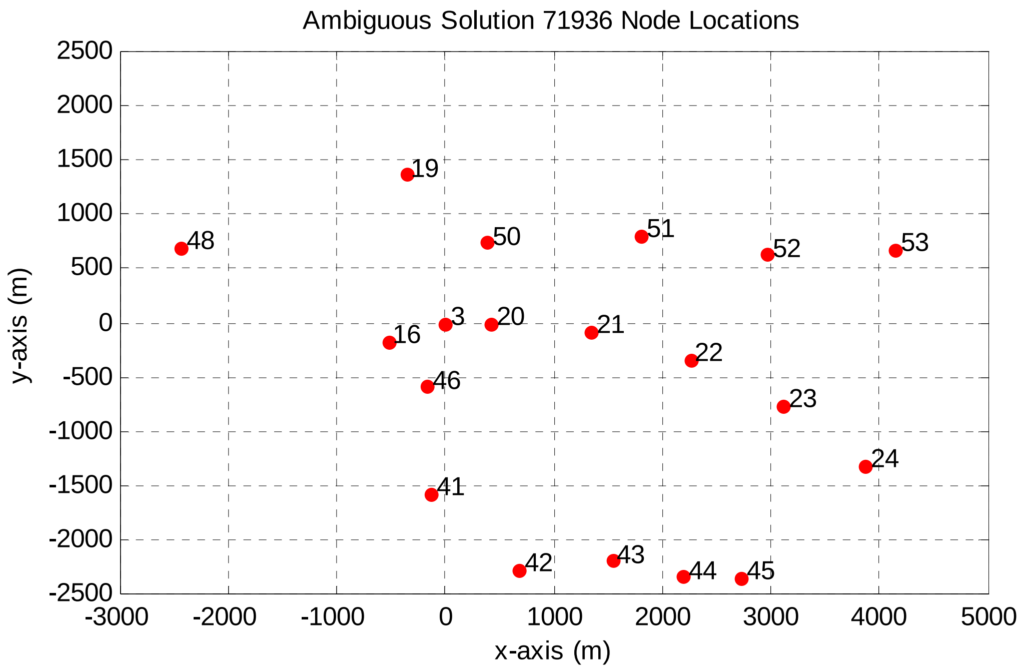
Matrix_Approach_3.m

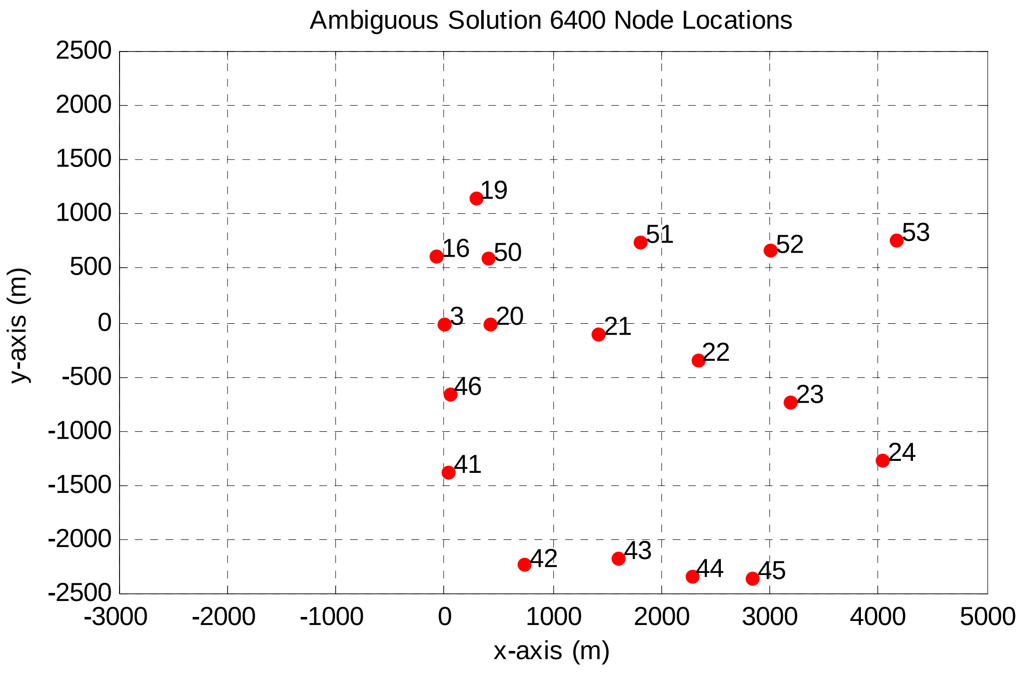
```
function change = Matrix_Approach_3(n)

%LT Dave Zinkhon    22 SEP 08
%Program to solve local position when 3 ranges are known to nodes with
        %know positions
%23 SEP 08 - Revised to make function that is passed ranges and current
local positions
                %and make changes for current node of interest (n is
index of node of interest)
%01 Oct 08 - Revised to take nodes and ranges 3 at a time and solve
vice
                %one single solution
%17 Oct 08 - Revised to use X_positions and Y_positions

global X_positions;
global Y_positions;
global ranges;
global active_branches;

%length is determined prior to executing loop to prevent unnecessary
    %looping if active_branches is added to during loop
length_active_branches = length(active_branches);

%set convergence threshold for solutions
conv_thresh = 17;


for v = 1:length_active_branches
%%
%points to proper column for given branch number
    branch_col = active_branches(v)+2;      %+2 aligns to proper column
number

%%
%Logic tests to determine which nodes are being referenced
    R = isnan(ranges(:,n));%logic test for NaN's in ranges
    LN = isnan(X_positions(:,branch_col));%logic test for NaN's in
nodes
    corr = R+LN;    %if result is zero, means is range to known
position

    usable = find (corr==0);    %array that know which nodes are usable
%%
if length(usable)>=3

%used to track when changes are made to position matrixes
    prior_x = X_positions (n,branch_col);
    prior_y = Y_positions (n,branch_col);
 %%
    combo = comb_gen(length(usable),3); %matrix of possible
combinations of usable data (see Reed)
```

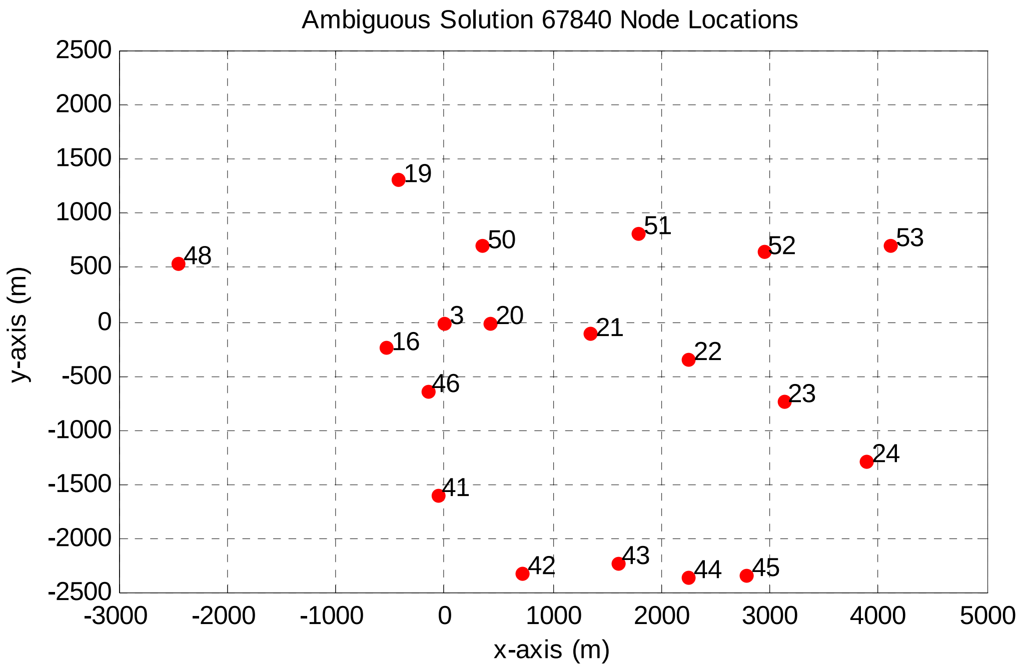
```
    for i = 1:length(combo(:,1))
        x = combo(i,1);     %x,y,z, are holders for which line of
usable should be referenced
        y = combo(i,2);
        z = combo(i,3);

        k = 1;
        for j = [x,y,z]    %imports range data into proper matrix form
            a(k,1) = ranges (usable(j), n);
            c(k,1) = X_positions(usable(j),branch_col);
            c(k,2) = Y_positions(usable(j),branch_col);
            k=k+1;
        end

%Calculations performed here based on Reed's Matrix Formula
        sqr_mat = a(1:end-1, :).^2 - a(2:end, :).^2;
        b = sum(sqr_mat, 2);    %diff of sqr of ranges

        sqr_mat = c(1:end-1, :).^2 - c(2:end, :).^2;
        d = sum(sqr_mat, 2);    %diff of sqr of positions
        e = c(2:end, :) - c(1:end-1, :);    %diff of positions
        out = 2*e \ (b-d);


%verify that range from calculated position to accepted position are
        %within acceptable criteria
        poss (i,1) = out (1,1);
        poss (i,2) = out (2,1);
        accept_criteria = .1;   %fraction difference between actual
range
                                %and calculated range that is
acceptable
        for p = 1:3
            range_chk = sqrt((poss(i,1)-c(p,1))^2+(poss(i,2)-
c(p,2))^2);
            if abs((range_chk - a(p))/a(p)) > accept_criteria
                poss(i,1) = NaN;
                poss(i,2) = NaN;
            end
        end

    end

%take mean position of all combinations that were not rejected
    q = isnan(poss);    %logic test for good data

    sum_x = 0;  %used to take mean
    sum_y = 0;
    tot = 0;

    for r = 1:length(q(:,1))
        if q(r,1) == 0
            sum_x = sum_x + poss (r,1);
            sum_y = sum_y + poss (r,2);
            tot = tot + 1;
        end
    end

    X_positions (n, branch_col) = sum_x/tot;
    Y_positions (n, branch_col) = sum_y/tot;
%%
%test to see if change was made to local program
    if abs(prior_x - X_positions (n,branch_col)) <conv_thresh &&
abs(prior_y - Y_positions (n,branch_col))<conv_thresh
        change = 0;
    else
        change = 1;
    end

end

end
```

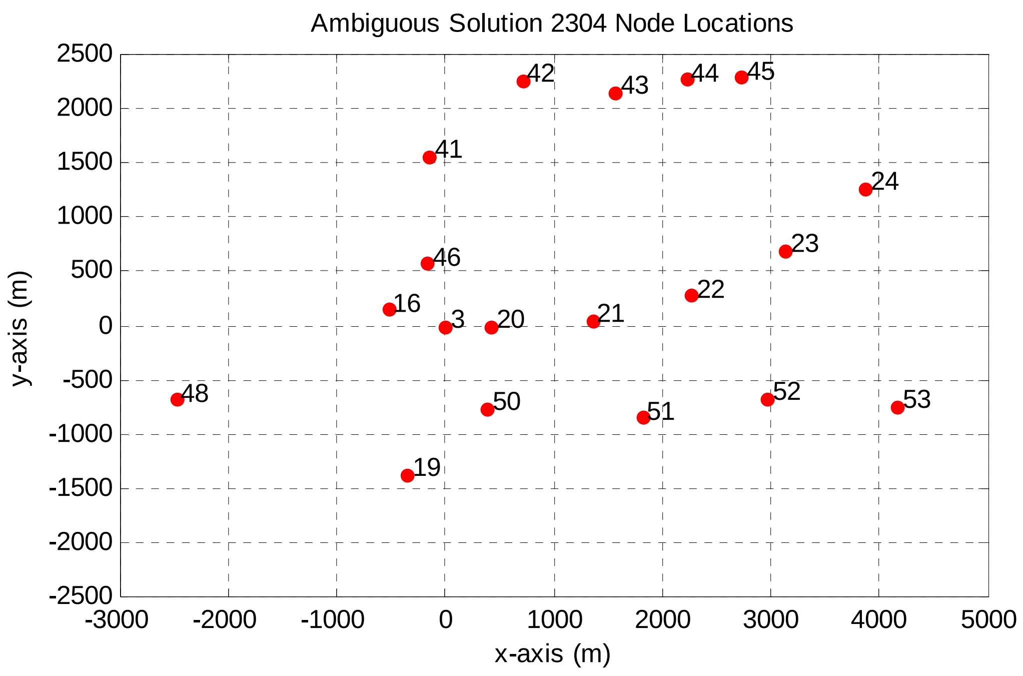
APPENDIX B: AMBIGUOUS SOLUTIONS FOR ST. MARGARET'S BAY SEA TRIAL

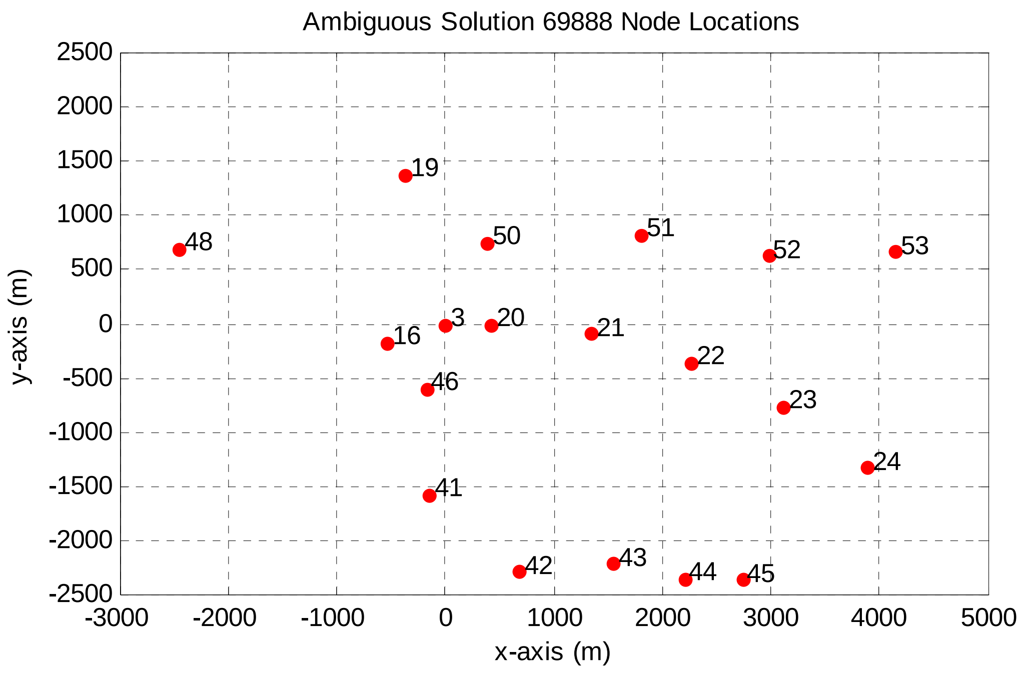
This appendix includes the sixteen ambiguous solutions for the ad hoc discoveries conducted at St. Margaret's Bay.



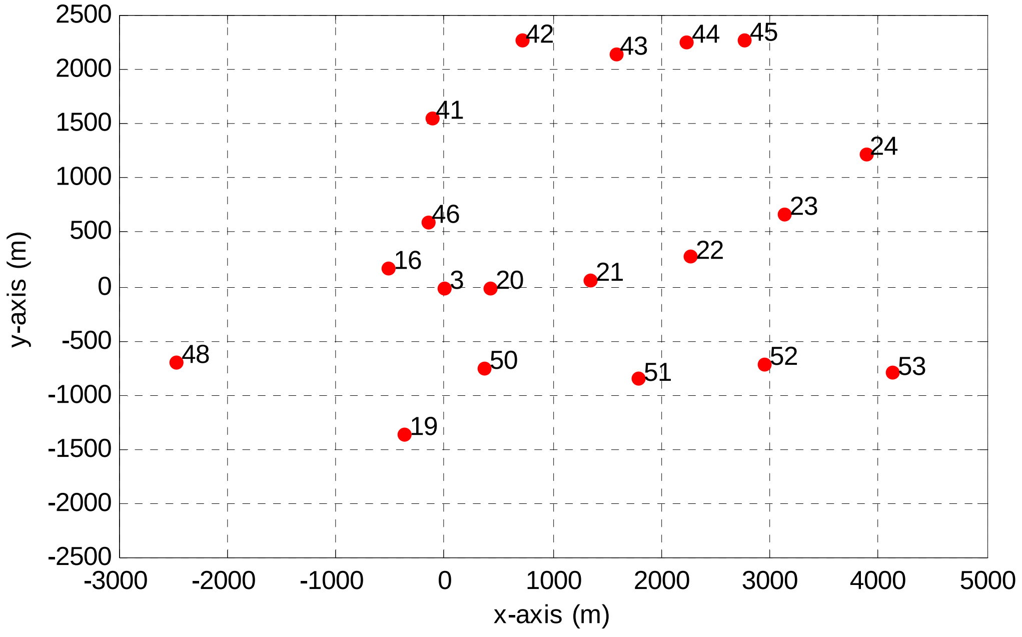

Ambiguous Solution 6400 Node Locations
y-axis (m)
x-axis (m)
2500
2000
1500
1000
500
0
-500
-1000
-1500
-2000
-2500
-3000
-2000
-1000
0
1000
2000
3000
4000
5000
19
16
50
51
52
53
3
20
21
22
23
46
24
41
42
43
44
45


Ambiguous Solution 67840 Node Locations
y-axis (m)
x-axis (m)
2500
2000
1500
1000
500
0
-500
-1000
-1500
-2000
-2500
-3000
-2000
-1000
0
1000
2000
3000
4000
5000
19
48
50
51
52
53
3
20
16
21
22
46
23
24
41
42
43
44
45

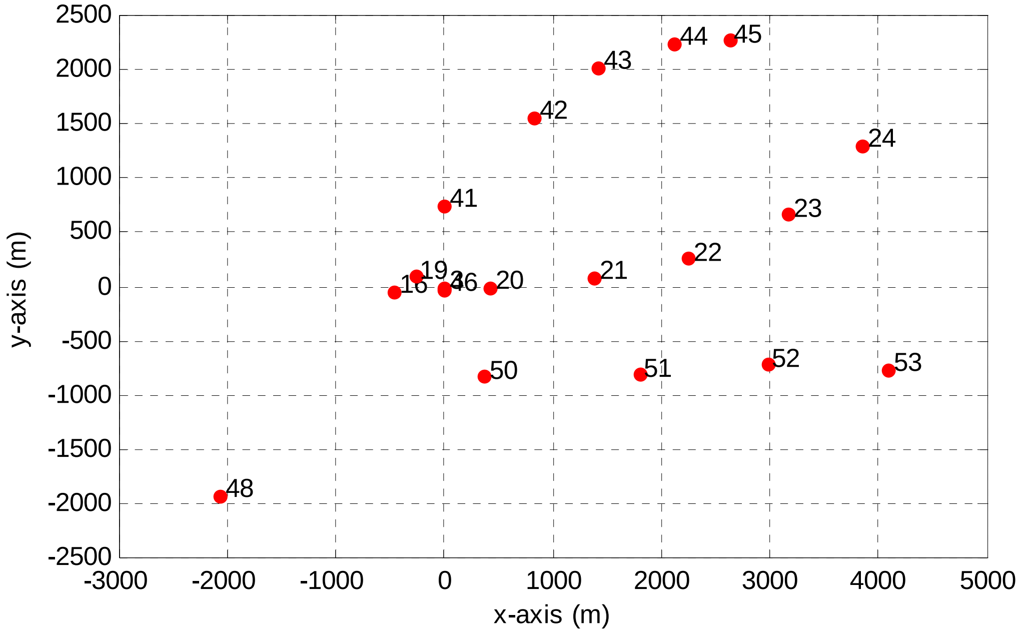

Ambiguous Solution 2304 Node Locations
y-axis (m)
x-axis (m)
-3000
-2000
-1000
0
1000
2000
3000
4000
5000
2500
2000
1500
1000
500
0
-500
-1000
-1500
-2000
-2500
42
43
44
45
41
24
23
46
22
16
3
20
21
48
50
51
52
53
19

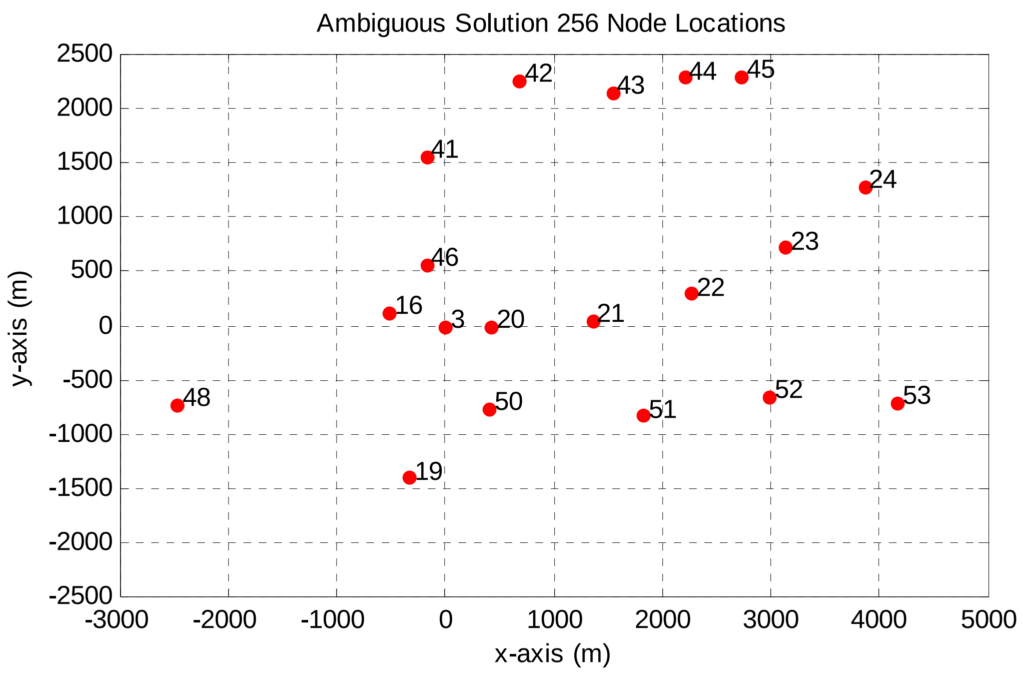

Ambiguous Solution 69888 Node Locations
y-axis (m)
x-axis (m)
-3000
-2000
-1000
0
1000
2000
3000
4000
5000
2500
2000
1500
1000
500
0
-500
-1000
-1500
-2000
-2500
19
48
50
51
52
53
3
20
16
21
22
46
23
24
41
42
43
44
45

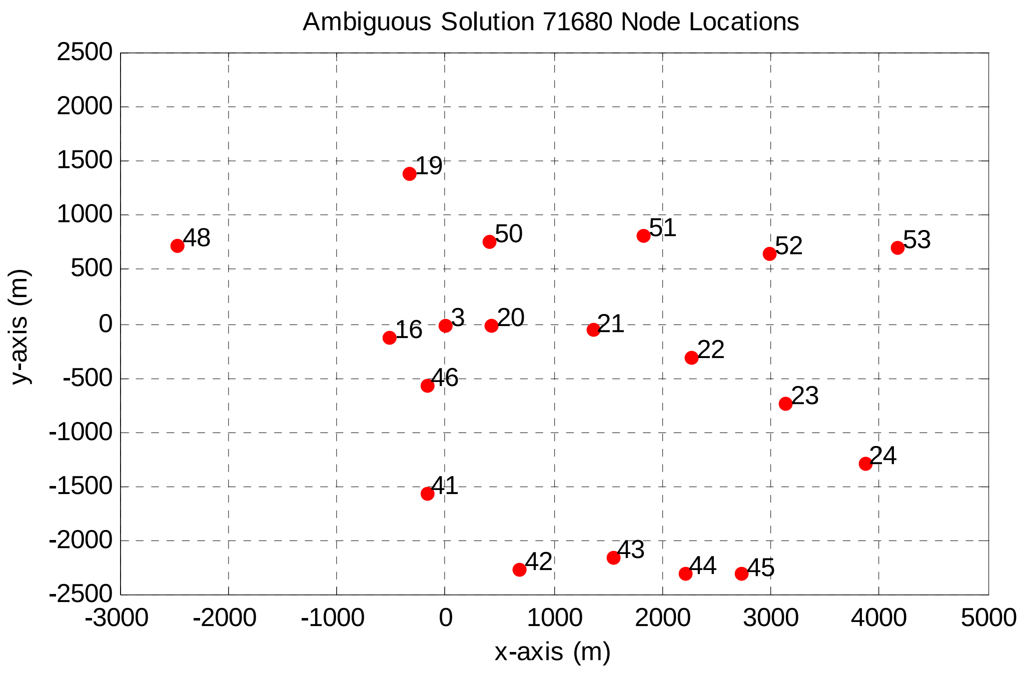
Ambiguous Solution 4352 Node Locations

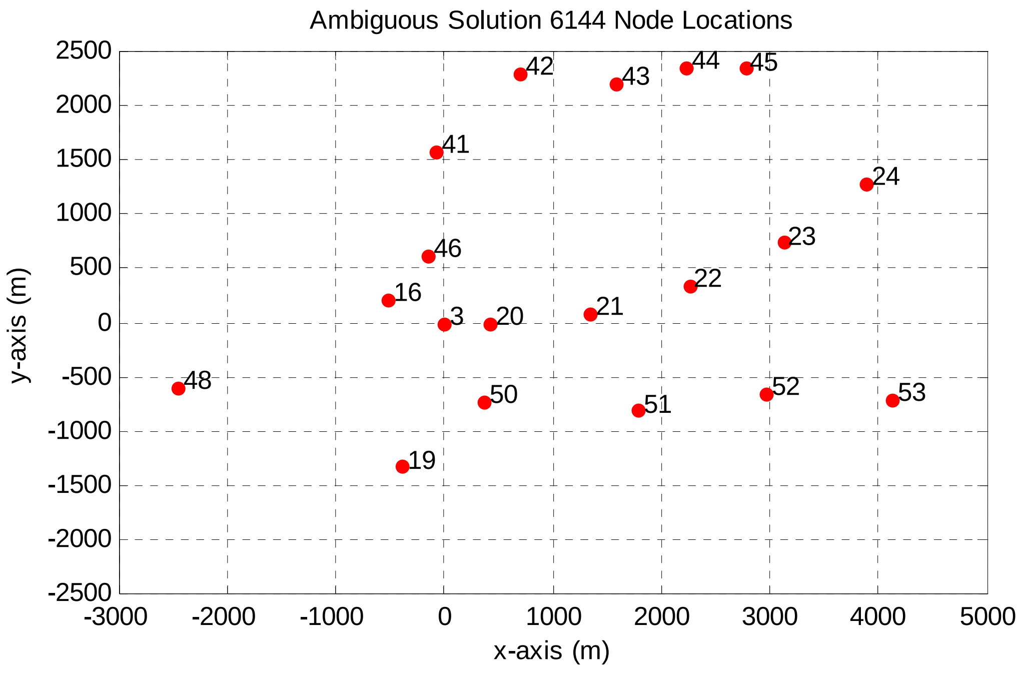


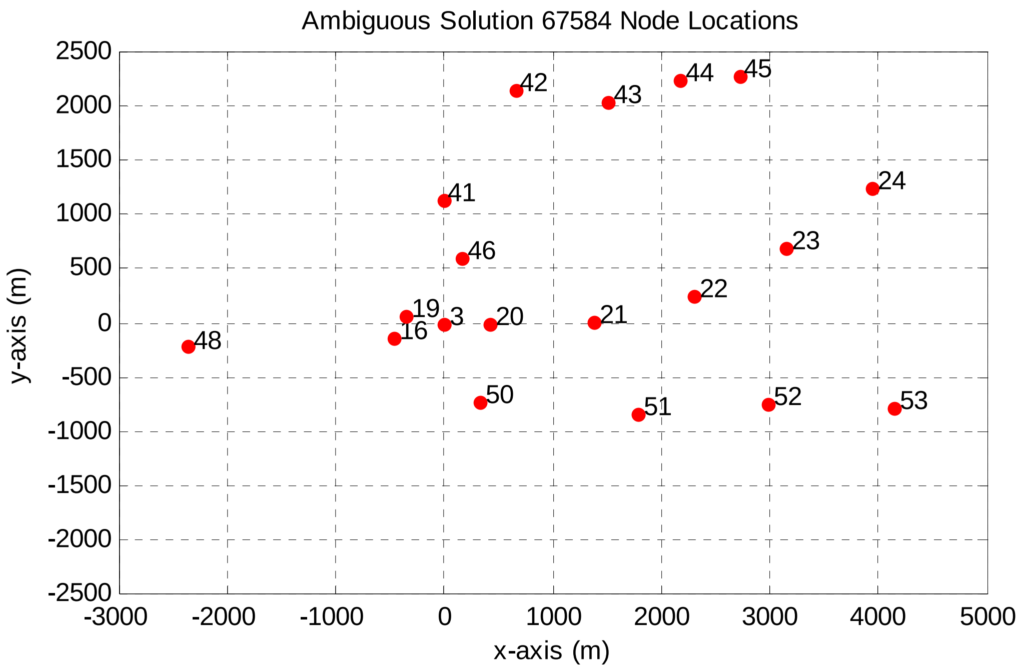
Ambiguous Solution 65792 Node Locations

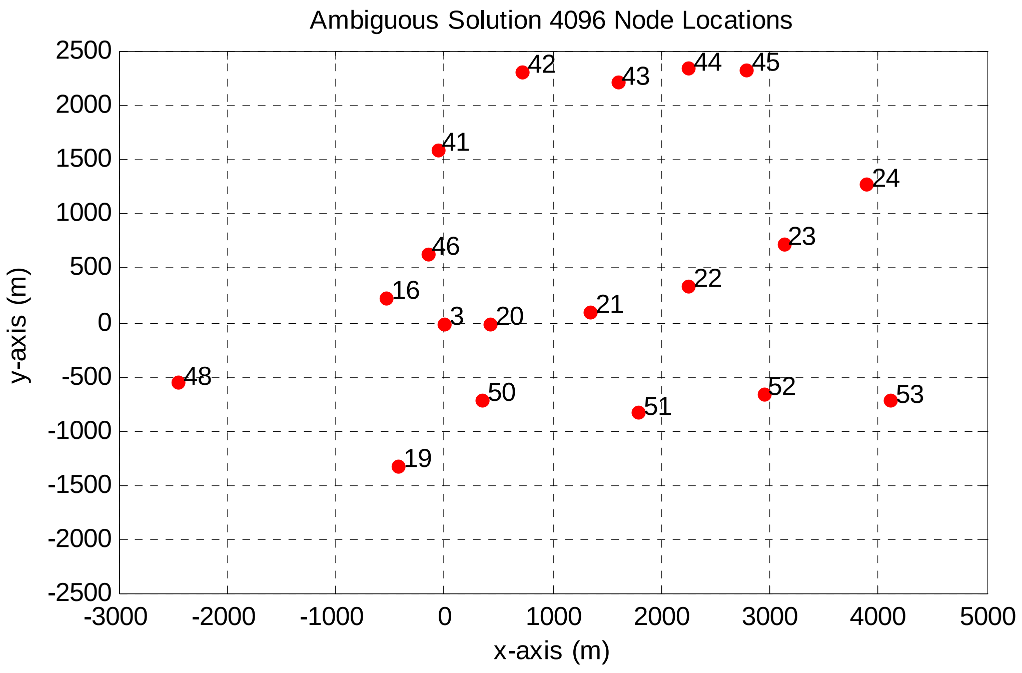


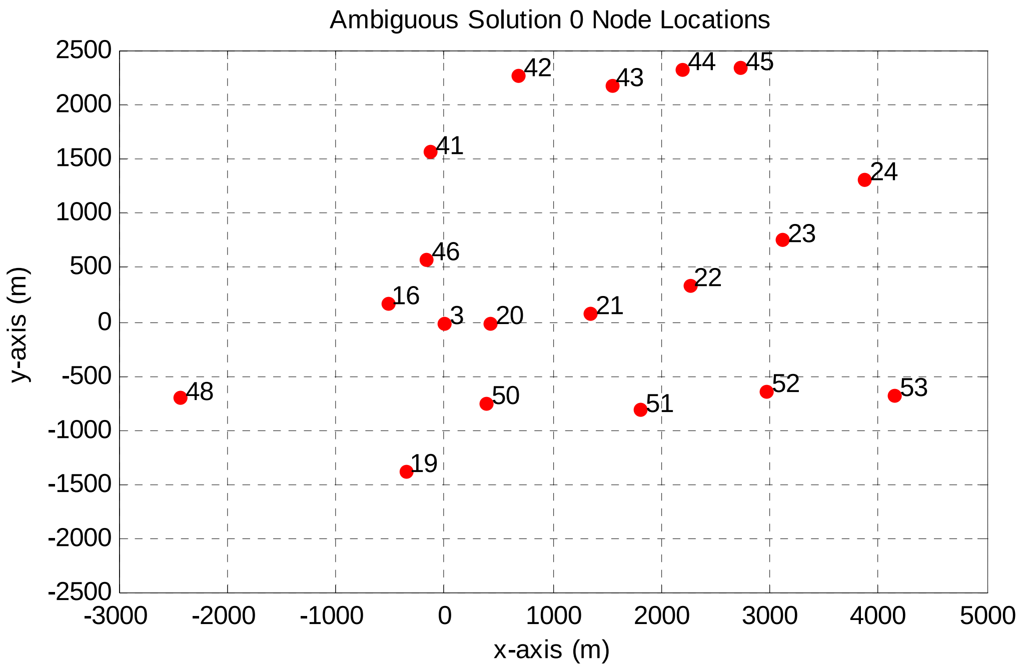

Ambiguous Solution 256 Node Locations
y-axis (m)
x-axis (m)
2500
2000
1500
1000
500
0
-500
-1000
-1500
-2000
-2500
-3000
-2000
-1000
0
1000
2000
3000
4000
5000
42
43
44
45
41
24
23
46
22
16
3
20
21
48
50
51
52
53
19

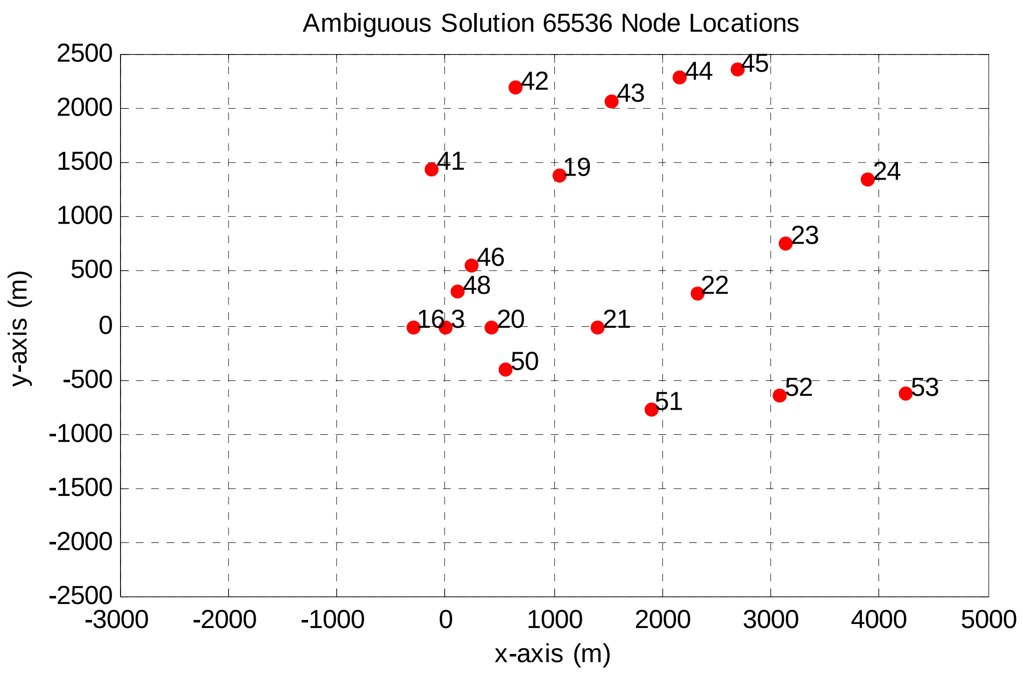

Ambiguous Solution 71680 Node Locations
y-axis (m)
x-axis (m)
2500
2000
1500
1000
500
0
-500
-1000
-1500
-2000
-2500
-3000
-2000
-1000
0
1000
2000
3000
4000
5000
19
48
50
51
52
53
3
20
16
21
22
46
23
24
41
43
42
44
45

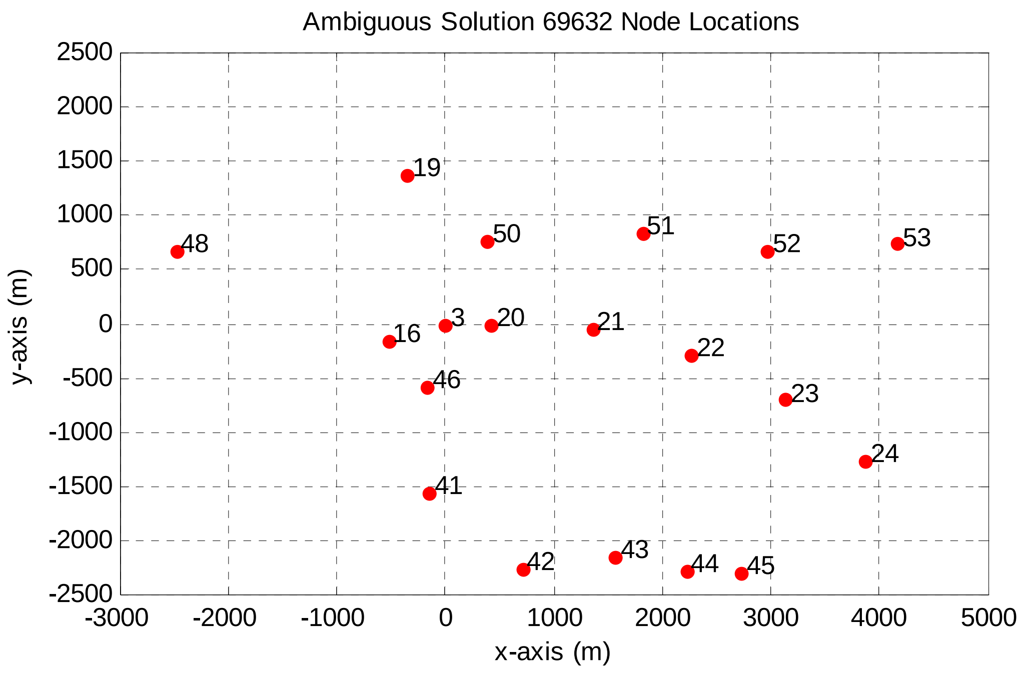

Ambiguous Solution 6144 Node Locations
y-axis (m)
x-axis (m)
2500
2000
1500
1000
500
0
-500
-1000
-1500
-2000
-2500
-3000
-2000
-1000
0
1000
2000
3000
4000
5000
42
43
44
45
41
24
23
46
22
16
3
20
21
48
52
53
50
51
19

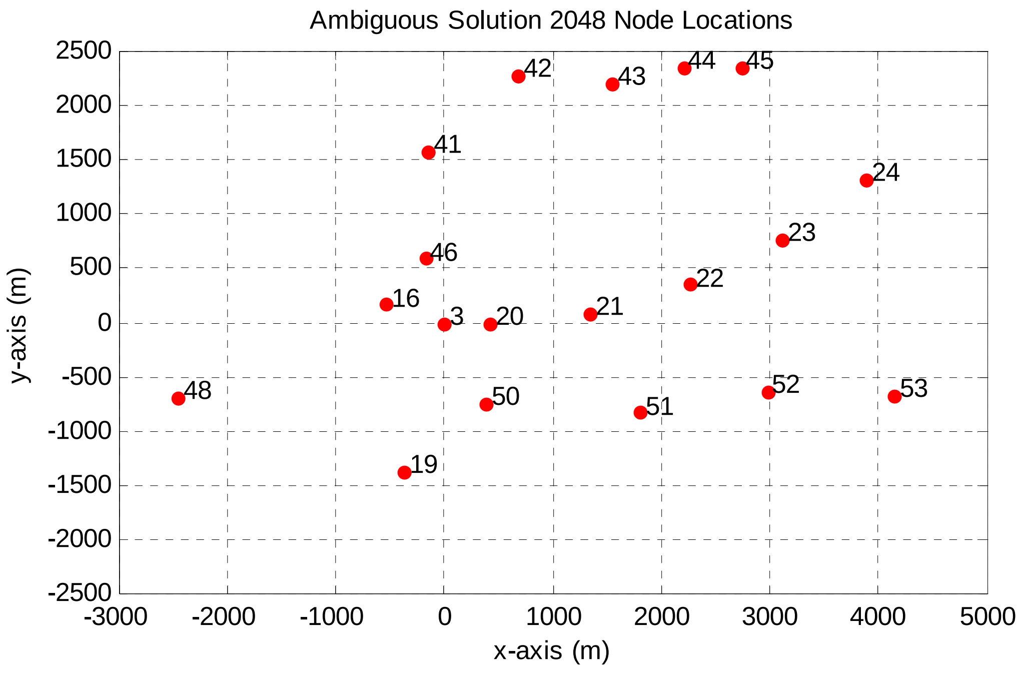

Ambiguous Solution 67584 Node Locations
y-axis (m)
x-axis (m)
2500
2000
1500
1000
500
0
-500
-1000
-1500
-2000
-2500
-3000
-2000
-1000
0
1000
2000
3000
4000
5000
42
43
44
45
41
24
46
23
22
19
3
20
21
48
16
50
51
52
53


Ambiguous Solution 4096 Node Locations
y-axis (m)
x-axis (m)
2500
2000
1500
1000
500
0
-500
-1000
-1500
-2000
-2500
-3000
-2000
-1000
0
1000
2000
3000
4000
5000
42
43
44
45
41
24
23
46
22
16
21
3
20
48
52
53
50
51
19


Ambiguous Solution 0 Node Locations
y-axis (m)
x-axis (m)
2500
2000
1500
1000
500
0
-500
-1000
-1500
-2000
-2500
-3000
-2000
-1000
0
1000
2000
3000
4000
5000
42
43
44
45
41
24
23
46
22
16
21
3
20
48
52
53
50
51
19


Ambiguous Solution 65536 Node Locations
y-axis (m)
x-axis (m)
42
43
44
45
41
19
24
23
46
48
22
16
3
20
21
50
51
52
53


Ambiguous Solution 69632 Node Locations
y-axis (m)
x-axis (m)
19
48
50
51
52
53
3
20
21
16
22
46
23
24
41
42
43
44
45


Ambiguous Solution 2048 Node Locations
y-axis (m)
x-axis (m)
2500
2000
1500
1000
500
0
-500
-1000
-1500
-2000
-2500
-3000
-2000
-1000
0
1000
2000
3000
4000
5000
42
43
44
45
41
24
23
46
22
16
3
20
21
48
50
51
52
53
19

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] C. W. Ong, *A Discovery Process for Initializing Ad Hoc Underwater Acoustic Networks*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, December 2008.

[2] M. H. Hahn, *Undersea Navigation via a Distributed Acoustic Communications Network*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, USA, June 2005.

[3] S. P. Ouimet, *Undersea Navigation of a Glider UUV Using an Acoustic Communications Network*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, USA, September 2005.

[4] M. S. Reed, *Use of a Acoustic Network as an Underwater positioning System,* M.S. Thesis, Naval Postgraduate School, Monterey, CA, USA, June 2006.

[5] H. A. Kriewaldt, *Communications Performance of an Undersea Acoustic Wide-Area Network*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, March 2006.

[6] J. A. Rice, "Seaweb Acoustic Communication and Navigation Networks," *Proc. Conf. Underwater Acoustic Measurements: Technologies & Results*, Heraklion, Greece (2005).

[7] D. J. Grimmet, "Message Routing Criteria for Undersea Acoustic Communication Networks," *Proc. IEEE Oceans 2007 – Europe*, pp. 1-6, June 2007.

[8] J. A. Rice, B. Creber, C. Fletcher, P. Baxley, K. Rogers, K. McDonald, D. Rees, M. Wolf, S. Merriman, R. Mechio, J. Proakis, K. Scussel, D. Porta, J. Baker, J. Hardiman, and D. Green, "Evolution of Seaweb Underwater Acoustic Networking," *Oceans 2000 MTS/IEEE Conference and Exhibition*, vol. 3, pp. 2007-2017, September 2000.

[9] J. A. Rice, V. K. McDonald, M. D. Green, and D. Ports, "Adaptive Modulation for Undersea Acoustic Telemetry," *Sea Technology*, vol. 40, no. 5, pp. 29-36, May 1999.

[10] J. Kalscheuer, *A Selective Automatic Repeat Request Protocol for Undersea Acoustic Links,* M.S. Thesis, Naval Postgraduate School, Monterey, CA, June 2004.

[11] S.P. Ouimet, M. J. Hahn, and J. A. Rice, "Undersea Communication Network as a UUV Navigation Aid," *Proc. IEEE Oceans 2005,* vol. 3, pp. 2485-2490, 2005.

[12] N. Bowditch, *The American Practical Navigator*, 2002 Bicentennial ed., Bethesda, MD. National Imagery and Mapping Agency, 2002.

[13] W. H. Munk, *Sound Channel in an exponentially stratified ocean, with application to SOFAR, Institute of Geophysics and Planetary Physics,* Scripps Institution of Oceanography, 1974.

[14] R. J. Urick, *Principles of Underwater Sound*, 3rd Edition, New York, NY. McGraw-Hill Book Company, 1983.

[15] J. A. Rice, *A Prototype Array-element Localization Sonobuoy*, Naval Ocean Systems Center, San Diego, CA. Report No. TR1365, December 1990.

[16] L. O. Krause, "A Direct Solution to GPS-Type Navigation Equations," *IEEE Transactions on Aerospace and Electric Systems*, vol, AES-23, no. 2, March 1987, pp. 225-232.

[17] R. E. Francois and G. R. Garrison, "Sound Absorption based on Ocean Measurements: Part I: Pure Water and Magnesium Sulfate Contributions," *Journal of the Acoustical Society of America*, vol. 72, no. 3, pp. 896-907, 1982.

[18] R. E. Francois and G. R. Garrison, "Sound Absorption based on Ocean Measurements: Part II: Boric Acid Contribution and Equation for Total Absorption," *Journal of the Acoustical Society of America*, vol. 72, no. 6, pp. 1879-1890, 1982.

[19] R. F. W. Coates, U*nderwater Acoustic Systems*, New York: Halsted Press, 1989.

[20] J. C. Torres, *Modeling of High-Frequency Acoustic Propagation in Shallow Water,* M.S. Thesis, Naval Postgraduate School, Monterey, CA, USA, June 2007.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Joseph A. Rice
   Naval Postgraduate School
   Monterey, California

4. RADM (Ret) Winford Ellis
   Naval Postgraduate School
   Monterey, California

5. RADM (Ret) Rick Williams
   Naval Postgraduate School
   Monterey, California

6. LCDR Vicki Taber
   Naval Postgraduate School
   Monterey, California

7. Professor Daphne Kapolka
   Naval Postgraduate School
   Monterey, California

8. Professor John Colosi
   Naval Postgraduate School
   Monterey, California

9. Professor Don Brutzman
   Naval Postgraduate School
   Monterey, California

10. Dana Hesse
    ONR 321 MS
    Arlington, Virginia

11. Doug Grimmett
    SPAWAR System Center Pacific
    San Diego, California

12. Bill Marn
SPAWAR System Center Pacific
San Diego, California

13. Chris Fletcher
SPAWAR System Center Pacific
San Diego, California

14. Bob Creber
SPAWAR System Center Pacific
San Diego, California

15. Garry Heard
Defence Research and Development Canada, Atlantic
Halifax, Nova Scotia, Canada

16. Roald Otnes
Forsvarets Forskningsinstitutt
Horten, Norway

17. LTC Ong Chee Wei
Republic of Singapore Navy
Singapore

18. LCDR Bjorn Kerstens
Defence Materiel Organisation
The Hague, Netherlands

19. LT Scott Thompson
Naval Postgraduate School
Monterey, California

20. LT Jeremy Biediger
Naval Postgraduate School
Monterey, California

21. LT David Zinkhon
United States Navy
Groton, Connecticut